

日 本 国 特 許 庁
JAPAN PATENT OFFICE

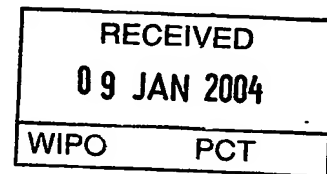
03.10.03

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: 2002年10月 3日

出 願 番 号
Application Number: 特願2002-291708
[ST. 10/C]: [JP2002-291708]



出 願 人
Applicant(s): 株式会社インフォーエス

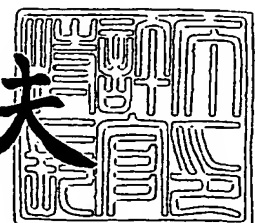
BEST AVAILABLE COPY

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

2003年12月18日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫



【書類名】 特許願

【整理番号】 020320P531

【あて先】 特許庁長官殿

【国際特許分類】 H04L 12/00

【発明者】

 【住所又は居所】 東京都多摩市鶴牧 6 丁目 1 6 番地 4 - 5 0 2

 【氏名】 佐藤 哲朗

【発明者】

 【住所又は居所】 東京都練馬区上石神井 2 丁目 6 番 1 2 号

 【氏名】 河口 文法

【特許出願人】

 【識別番号】 502254981

 【氏名又は名称】 有限会社ディシーエル

【代理人】

 【識別番号】 100102934

 【弁理士】

 【氏名又は名称】 今井 彰

【手数料の表示】

 【予納台帳番号】 050728

 【納付金額】 21,000円

【提出物件の目録】

 【物件名】 明細書 1

 【物件名】 図面 1

 【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 ビットストリングの照合方法

【特許請求の範囲】

【請求項1】 被検索ビットストリングを複数ビットの部分被検索ビットストリングに分けて、予め登録された複数の登録ビットパターンと多段階で照合するビットストリングの照合方法であって、1つの照合段階は、

前記被検索ビットストリングから、現段階の前記部分被検索ビットストリングを選択し、その部分被検索ビットストリングが取り得る全ての値と照合する全照合工程と、

前記登録ビットパターンの部分ビットパターンであって、先行する段階から得られる照合継続情報により決まる、前記部分被検索ビットストリングに対応する段階の部分登録ビットパターンを示す前記現段階のマスクデータを、前記全照合工程と前後して、または並列に、メモリよりロードするマスクロード工程と、

前記全照合工程の結果と前記現段階のマスクデータより、前記部分被検索ビットストリングに一致する前記部分登録ビットパターンの有無を少なくとも示す照合結果を得る判定工程と、

前記照合結果により、前記現段階のマスクデータに対応して記録された前記部分登録ビットパターンのそれぞれに続く次の段階のマスクデータのアドレスが示されたアドレステーブルの中から、前記一致する部分登録ビットパターンに続く次の段階のマスクデータのアドレスを含む前記照合継続情報を出力する出力工程とを有するビットストリングの照合方法。

【請求項2】 請求項1において、前記全照合工程とマスクロード工程とを並列に行う、ビットストリングの照合方法。

【請求項3】 請求項1において、前記判定工程では、前記部分被検索ビットストリングに値が最も近い最大または最小の前記登録部分ビットパターンの有無を示す照合結果が得られ、

前記出力工程では、前記一致する部分登録ビットパターンがないときに、前記アドレステーブルの中から、前記最大または最小の部分登録ビットパターンに続く次の段階のマスクデータのアドレスを含む前記照合継続情報を出力する、ビッ

トストリングの照合方法。

【請求項 4】 請求項 3 において、前記一致する部分登録ビットパターンがあり、前記最大または最小の部分登録ビットパターンがあるときに、前記最大または最小の部分登録ビットパターンに続く次の段階のマスクデータのアドレスを候補アドレスとして記憶する候補アドレス記憶工程を有し、

前記出力工程では、前記一致する部分登録ビットパターンおよび前記最大または最小の部分登録ビットパターンがないときに、前記候補アドレスを含む前記照合継続情報を出力する、ビットストリングの照合方法。

【請求項 5】 請求項 3 において、前記最大または最小の部分登録ビットパターンに続く前記現段階のマスクデータのアドレスを含む前記照合継続情報が与えられたときは、前記全照合工程および判定工程をバイパスする工程を有し、

前記出力工程では、前記現段階のマスクデータに対応する前記アドレステーブルから、前記現段階のマスクデータに示された最大または最小の部分登録ビットパターンに続く次の段階のマスクデータのアドレスを含む前記照合継続情報を出力する、ビットストリングの照合方法。

【請求項 6】 請求項 3 において、前記マスクデータのそれぞれに対して、そのマスクデータに示された最大また最小の部分登録ビットパターンに対応して決まる最大または最小の前記登録ビットパターンを示すバイパスデータが記憶されており、

前記最大または最小の部分登録ビットパターンに続く前記現段階のマスクデータのアドレスを含む前記照合継続情報が与えられたときは、前記全照合工程、マスキロード工程および判定工程をバイパスする工程と有し、

前記出力工程では、前記現段階のマスクデータに対応する前記バイパスデータを含む最終照合情報を出力して前記被検索ビットストリングの照合を終了する、ビットストリングの照合方法。

【請求項 7】 請求項 1 において、前記マスクデータおよびアドレステーブルを更新することにより、前記登録ビットパターンを追加または削除する更新工程を有する、ビットストリングの照合方法。

【請求項 8】 請求項 1 において、複数の前記照合段階がパイプライン方式

で実行される、ビットストリングの照合方法。

【請求項 9】 請求項 1 において、複数の前記照合段階の間では、前記照合継続情報を含むデータがパケット化されて伝達される、ビットストリングの照合方法。

【請求項 10】 請求項 3 において、複数の前記照合段階を備えた分類過程を複数有し、前記最大または最小の部分登録ビットパターンから導かれる最大または最小の前記登録ビットパターンは複数の分類結果を含む範囲を示しており、前記分類過程により得られる前記複数の分類結果の論理積を演算し、最終の分類結果を得る論理演算工程を有する、ビットストリングの照合方法。

【請求項 11】 請求項 10 において、前記論理演算工程では、前記複数の分類結果の論理積を行列演算し、最終の分類結果を得る、ビットストリングの照合方法。

【請求項 12】 請求項 1 に記載の照合方法により、前記被検索ビットストリームを照合する工程と、

その結果に基づき前記被検索ビットストリングを備えたデータを管理する工程とを有するデータ管理方法。

【請求項 13】 請求項 10 に記載の照合方法により、IP アドレス、ポート番号およびプロトコルの少なくともいずれかを含むパケットを管理するためのビットストリングを前記被検索ビットストリングとし、それが属する前記範囲を照合する工程と、

その結果に基づき前記被検索ビットストリングを備えた前記パケットを管理する工程とを有するデータ管理方法。

【請求項 14】 被検索ビットストリングを複数ビットの部分被検索ビットストリングに分けて、予め登録された複数の登録ビットパターンと多段階で照合するための 1 つの照合段階を実行する照合装置であって、

前記被検索ビットストリングから、現段階の前記部分被検索ビットストリングを選択し、その部分被検索ビットストリングが取り得る全ての値と照合する全照合手段と、

前記登録ビットパターンの部分ビットパターンであって、先行する段階から得

られる照合継続情報により決まる、前記部分被検索ビットストリングに対応する段階の部分登録ビットパターンを示す前記現段階のマスクデータをメモリよりロードするマスクロード手段と、

前記全照合手段の結果と前記現段階のマスクデータより、前記部分被検索ビットストリングに一致する前記部分登録ビットパターンの有無を少なくとも示す照合結果を出力する判定手段と、

前記照合結果により、前記現段階のマスクデータに対応して記録された前記部分登録ビットパターンのそれぞれに続く次の段階のマスクデータのアドレスが示されたアドレステーブルの中から、前記一致する部分登録ビットパターンに続く次の段階のマスクデータのアドレスを含む前記照合継続情報を出力する出力手段とを有する照合装置。

【請求項 15】 請求項 14 において、前記判定手段は、前記部分被検索ビットストリングに値が最も近い最大または最小の前記登録部分ビットパターンの有無を示す照合結果を出力し、

前記出力手段は、前記一致する部分登録ビットパターンがないときに、前記アドレステーブルの中から、前記最大または最小の部分登録ビットパターンに続く次の段階のマスクデータのアドレスを含む前記照合継続情報を出力する、照合装置。

【請求項 16】 請求項 15 において、前記出力手段は、前記一致する部分登録ビットパターンがあり、前記最大または最小の部分登録ビットパターンがあるときに、前記最大または最小の部分登録ビットパターンに続く次の段階のマスクデータのアドレスを候補アドレスとして出力し、さらに、

前記一致する部分登録ビットパターンおよび前記最大または最小の部分登録ビットパターンがないときに、前記候補アドレスを含む前記照合継続情報を出力する、照合装置。

【請求項 17】 請求項 15 において、前記出力手段は、前記最大または最小の部分登録ビットパターンに続く前記現段階のマスクデータのアドレスを含む前記照合継続情報が与えられたときは、前記現段階のマスクデータに対応する前記アドレステーブルから、前記現段階のマスクデータに示された最大または最小

の部分登録ビットパターンに続く次の段階のマスクデータのアドレスを含む前記照合継続情報を出力する、照合装置。

【請求項 18】 請求項 15 において、前記マスクデータのそれぞれに対して、そのマスクデータに示された最大また最小の部分登録ビットパターンに対応して決まる最大または最小の前記登録ビットパターンを示すバイパスデータが記憶されており、

前記出力手段は、前記最大または最小の部分登録ビットパターンに続く前記現段階のマスクデータのアドレスを含む前記照合継続情報が与えられたときは、前記現段階のマスクデータに対応する前記バイパスデータを最終照合情報として出力する、照合装置。

【請求項 19】 請求項 14 において、前記マスクデータおよびアドレステーブルを更新することにより、前記登録ビットパターンを追加または削除する更新手段を有する、照合装置。

【請求項 20】 請求項 14 に記載の照合装置を複数有する分類装置。

【請求項 21】 請求項 20 において、複数の前記照合装置はパイプラインで接続されている、分類装置。

【請求項 22】 請求項 20 において、複数の前記照合装置の間では、前記照合継続情報を含むデータがパケット化されて伝達される、分類装置。

【請求項 23】 請求項 20 において、前記メモリに対してデータを入出力するキャッシュ装置を有する、分類装置。

【請求項 24】 請求項 23 において、前記メモリには、前記マスクデータおよび前記アドレステーブルを含む照合テーブルが記憶されており、前記キャッシュ装置のキャッシュラインサイズが前記照合テーブルのサイズと同一である、分類装置。

【請求項 25】 請求項 24 において、最初の照合テーブルを指定する検索種別情報を受信する手段を有し、

前記キャッシュ装置は、それぞれの検索種別および前記照合段階の単位でキャッシュラインを割り当て、前記検索種別および照合段階の単位で前記キャッシュラインを管理する管理手段を備えている、分類装置。

【請求項 26】 請求項 24 において、最初の照合テーブルを指定する検索種別情報を受信する手段を有し、

前記メモリには、それぞれの検索種別および前記照合段階の単位で割り当てられた個別アドレス領域に前記照合テーブルが記憶されており、

前記キャッシュ装置のキャッシュラインは、前記個別アドレス領域単位で割り当てられている、分類装置。

【請求項 27】 請求項 23 において、前記キャッシュ装置は、照合テーブルがオンキャッシュでなければ前記照合装置にその旨を通知し、その照合装置の処理を中断して待ち行列に戻す、分類装置。

【請求項 28】 請求項 20 において、当該分類装置の分類結果を記憶した履歴キャッシュ装置であって、前記被検索ビットストリングを前記照合装置に供給する前に、当該分類装置の分類結果と前記被検索ビットストリングとを比較する履歴キャッシュ装置を有する分類装置。

【請求項 29】 請求項 20 に記載の分類装置を有し、前記最大または最小の部分登録ビットパターンから導かれる最大または最小の前記登録ビットパターンは複数の分類結果を含む範囲を示しており、

さらに、前記分類装置により得られる複数の分類結果の論理積を演算し、最終の分類結果を得る論理演算装置を有する検索装置。

【請求項 30】 請求項 29 において、複数の前記分類装置を有する検索装置。

【請求項 31】 請求項 29 において、前記論理演算装置は、前記複数の分類結果の論理積を行列演算し、最終の分類結果を得る、検索装置。

【請求項 32】 請求項 29 において、前記メモリに対してデータを入出力するキャッシュ装置を有する、検索装置。

【請求項 33】 請求項 29 において、当該検索装置の検索結果を記憶した履歴キャッシュ装置であって、前記被検索ビットストリングを前記分類装置に供給する前に、当該検索装置の検索結果と前記被検索ビットストリングとを比較する履歴キャッシュを有する検索装置。

【請求項 34】 請求項 20 に記載の分類装置と、

前記分類装置の出力に基づき前記被検索ビットストリングを備えたデータを管理する手段とを有するデータ管理装置。

【請求項 35】 請求項 29 に記載の検索装置を有し、

前記検索装置に、IP アドレス、ポート番号およびプロトコルの少なくともいずれかを含むパケットを管理するためのビットストリングを前記被検索ビットストリングとして供給する手段と、

前記検索装置の出力に基づき前記被検索ビットストリングを備えた前記パケットを管理する手段とを有するデータ管理装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、IP アドレスなどのビットストリングを検索あるいは照合に適した装置および方法に関するものである。

【0002】

【従来の技術】

インターネットなどのコンピュータネットワークを介してデータを送受信する際には所定のプロトコルが適用され、インターネットのネットワーク層におけるプロトコルはインターネットプロトコル (IP) と呼ばれている。この IP は情報をパケット化して所定の目的地まで移動させるが、そのために、情報を格納したペイロードに IP ヘッダをつけたパケットを送受信する。そして、IP ヘッダの情報に基づいて、各々のパケットは所定のルートに配送され、また、受信先では、IP ヘッダの情報に基づいてペイロードの情報が送信元の状態となるように組み立てられる。

【0003】

また、IP ヘッダの情報は、ルータおよびファイアウォールなどのネットワークを管理する装置あるいはソフトウェア (プログラムあるいはプログラム製品) で参照され、ルーティング、パケットフィルタリング、SPD (セキュリティ・ポリシーデータベース)、ポリシールーティング、QoS などの様々な目的のために照合され、パケットの送受信あるいは転送については常に利用される。

【0004】

さらに、今後、モバイルIPが実現されると、ルーティングテーブルは動的に変動する可能性があり、また、現状のIPv4（バージョン4）では、受信先アドレス（SIP）および送信元アドレス（DIP）は32ビットであるのに対し、IPv6（バージョン6）では128ビットに拡張される。したがって、今後、IPアドレスの検索における処理速度をさらに向上させることが、インターネットなどのコンピュータネットワークを用いてデータ交換をする上で非常に重要な技術となる。

【0005】

たとえば、特開平11-88427号公報には、ソートされた複数のエントリからなるルーティングテーブルに対して検索IPアドレスを大小比較しながら一致するIPアドレスを検索する技術が開示されているが、この方法によりルーティングあるいはその他の目的のために128ビットのIPアドレスを複数含むビットストリングを検索しようとするると膨大な時間が必要になり、実用的ではない。

【0006】

【特許文献1】

特開平11-88427号公報

【0007】

【発明が解決しようとする課題】

今後、IPアドレスに限らず、検索対象となるビットストリング（ビットストリームあるいはビットパターン）の情報量は増大する傾向にあり、光通信が本格化する場合は、Gビットクラスのビットストリングの検索機能が要求される。膨大なデータ量のビットストリングの検索方法は幾つか提案されている。まず、ソフトウェア方式は、検索機能は十分だが、速度的には不十分である。

【0008】

ハードウェアを用いた方式では、CAM（Content-Addressable Memory）方式がある。しかしながら、1個のCAMを使った繰り返し検索での最長一致検索、あるいは複数個のCAMを並列に使っての最長一致検索を実現するところまでし

かできず、パケットフィルタリングやステートフルインスペクションに要求される多次元範囲、たとえば、ソースIP・デスティネーションIP・ソースポート・デスティネーションポート・プロトコルを含む範囲の一致検索を実現するのは非常に困難である。また、最長一致検索の対象としてIPルーティングを考えた場合、CIDR形式での可変プレフィックス長への対応は、繰り返し型の繰り返し回数増や多数個CAM並列型でおこなうことになり、前者であればスループットやレイテンシ面での性能不足、後者であれば経路エントリの分布ばらつきにより、CAMエントリ利用率が悪くなるという問題点を抱える。

【0009】

登録エントリ数増に対しては複数CAMチップのカスケード接続でしか対応できない。被検索キーのビット長の増加に対しては、チップ自体のスペック変更でしか対応できず、全エントリのCAMメモリビット増が必要である。多チップ構成での対応の可能性については、CAMは多エントリ並列に照合されるため、複数チップ間で連動した照合をおこなうためには各エントリの照合状態をチップ間で継承しなければならない、CAMに求められる大量エントリ数を考えると現実的ではない。

【0010】

部分的にCAMを利用し、CAMにより検索対象エントリを絞り込み、その後パトリシアツリー等をソフトウェアにより検索するという方式もある。この方式ではツリー検索にソフトウェアを用いるため、完全ソフトウェア方式と同様の問題がある。また、ツリー検索部をハードウェア化した場合でも、例えばツリー構造がパトリシアであれば登録削除に複雑なツリー再構成が伴い、上位アプリケーションソフトウェアに過度なテーブル管理コストを強いることになる。

【0011】

マスキングが可能で、階層型の検索を可能としたターナリーCAM方式がある。この場合、ハードウェアそのものが提供する検索機能は、完全一致(CAM)と最長一致である。TCAMは原理的にはエントリのDC(ドントケア)ビットをも一致と見なす完全一致であり、照合部からは複数の一致エントリが出力される。そのため、照合部の後段にプレフィックス長あるいはエントリNo等による何

らかのプライオリティエンコードがないと、「複数ヒットから最長一致」の絞り込みをおこなえない。

【0012】

従って、TCAMにおいてはエントリの登録削除についての管理が煩雑となる。特にエントリNoがプライオリティとなる方式のTCAMの場合、エントリの追加にともない、既存エントリの再配置を上位アプリケーションソフトウェア側が自発的に行わねばならない。この既存エントリの再配置はプレフィックス長をキーとしたソートであり、複数回のエントリ位置交換が要求される。プレフィックス長だけが問題なので通常のソートよりは低コストであるが、ソートそのものは排除できない。すなわち、TCAMで範囲一致は実現できるが、それはエントリ管理をおこなう上位アプリケーションソフトウェアによるものであり、範囲表現を複数の最長一致表現へと分割し、最長一致へ還元して実現することになる。そのため、範囲一致のルール管理（削除・登録）が煩雑となる。つまり、TCAMにおける範囲一致は、静的なエントリからなる検索については実用性があったとしても、動的に変化するエントリからなる検索については、かなり上位アプリケーションソフトウェアへの負荷が高いといえる。あるいは、これら複雑なテーブル管理をおこなう専用のコプロセッサをTCAM自体とは別途用意することになる。

【0013】

以上のことから、範囲一致は無論、最長一致であっても、登録エントリがダイナミックに追加削除されていく性質のある検索にはTCAMは向かないといえる。登録エントリ数増に対しては複数のTCAMチップのカスケード接続でしか対応できないことはCAMと同様である。被検索キーのビット長の増加に対しても、チップ自体のスペック変更でしか対応できず、全エントリのTCAMメモリビット増が必要である。多チップ構成での対応の可能性については、TCAMは多エントリ並列に照合されるため、複数チップ間で連動した照合をおこなうためには各エントリの照合状態をチップ間で継承しなければならない、TCAMに求められる大量エントリ数を考えると現実的ではない。

【0014】

そこで、本発明においては、範囲一致を始め、完全一致および最長一致といった検索機能を提供可能であり、パケットフィルタリングやステートフルインスペクションにおいても上位アプリケーションソフトウェアの負担が軽いビットストリングの照合方法および照合装置を提供することを目的としている。特に、エントリの追加削除にともなうエントリ移動による、上位アプリケーションによるテーブル管理のオーバーヘッドを縮小または排除できるビットストリングの照合方法および照合装置を提供することを目的としている。

【0015】

さらに、メモリとして、SDRAM等通常のDRAM（あるいはSRAM）を使用可能とし、十分な検索速度を提供しながら、CAM系の専用メモリよりもはるかに廉価で大容量を実現できる照合方法および装置を提供することも本発明の目的である。そして、そのような照合装置は、収容可能なエントリ数はSDRAM容量にのみ規定されるため、スケーラブル性が高い。また、メモリ管理が柔軟に行えるため、複数種の検索を単一のハードウェアセットで柔軟におこなえるものになる。また、検索対象となる被検索ビットストリング（キー）のビット長増に対してハードウェア自体のアルゴリズム的な変更は必要なく、ユニットを追加したり、一連の処理を繰り返すことで対応可能な照合方法および照合装置を提供することも目的としている。

【0016】

【課題を解決するための手段】

本発明のビットストリングの照合方法においては、被検索ビットストリングを複数ビットの部分被検索ビットストリングに分けて、予め登録された複数の登録ビットパターンと多段階で照合する方法を採用する。そして、1つの照合段階は、被検索ビットストリングから、現段階の部分被検索ビットストリングを選択し、その部分被検索ビットストリングが取り得る全ての値と照合する全照合工程と、登録ビットパターンの部分ビットパターンであって、先行する段階から得られる照合継続情報により決まる、部分被検索ビットストリングに対応する段階の部分登録ビットパターンを示す現段階のマスクデータを、全照合工程と前後して、または並列に、メモリよりロードするマスクロード工程と、全照合工程の結果と

現段階のマスクデータより、部分被検索ビットストリングに一致する部分登録ビットパターンの有無を少なくとも示す照合結果を得る判定工程と、照合結果により、現段階のマスクデータに対応して記録された部分登録ビットパターンのそれぞれに続く次の段階のマスクデータのアドレスが示されたアドレステーブルの中から、一致する部分登録ビットパターンに続く次の段階のマスクデータのアドレスを含む照合継続情報を出力する出力工程とを有する。

【0017】

この照合方法を実現する照合装置として、本発明においては、被検索ビットストリングを複数ビットの部分被検索ビットストリングに分けて、予め登録された複数の登録ビットパターンと多段階で照合するための1つの照合段階を実行する照合装置を提供する。この照合装置は、被検索ビットストリングから、現段階の部分被検索ビットストリングを選択し、その部分被検索ビットストリングが取り得る全ての値と照合する全照合手段と、登録ビットパターンの部分ビットパターンであって、先行する段階から得られる照合継続情報により決まる、部分被検索ビットストリングに対応する段階の部分登録ビットパターンを示す現段階のマスクデータをメモリよりロードするマスクロード手段と、全照合手段の結果と現段階のマスクデータより、部分被検索ビットストリングに一致する部分登録ビットパターンの有無を少なくとも示す照合結果を出力する判定手段と、照合結果により、現段階のマスクデータに対応して記録された部分登録ビットパターンのそれぞれに続く次の段階のマスクデータのアドレスが示されたアドレステーブルの中から、一致する部分登録ビットパターンに続く次の段階のマスクデータのアドレスを含む照合継続情報を出力する出力手段とを有する。

【0018】

これらの照合方法および照合装置は、完全一致および最長一致の検索機能を提供する。さらに、判定工程および判定手段において、部分被検索ビットストリングに値が最も近い最大または最小の登録部分ビットパターンの有無を示す照合結果が得られると、出力工程および出力手段では、一致する部分登録ビットパターンがないときに、アドレステーブルの中から、最大または最小の部分登録ビットパターンに続く次の段階のマスクデータのアドレスを含む照合継続情報を出力す

ることにより、検索機能として範囲一致を含めた照合方法および照合装置を提供できる。ここで、部分被検索ビットストリングに値が最も近い最大または最小の部分登録ビットパターンとは、部分被検索ビットストリングより値が小さくて最大、または値が大きくて最小の部分登録ビットパターンを意味する。最大の部分登録ビットパターンに続くアドレスが選択されることにより、以下で説明する右倒れの範囲検索が可能となる。また、最小の部分登録ビットパターンに続くアドレスが選択されることにより、以下で説明する左倒れの範囲検索が可能となる。したがって、右倒れ型の検索においては、以下で説明する最大の側が選択され、左倒れ型の検索においては、以下で説明する最小の側が選択される。

【0019】

また、本発明の照合方法および照合装置においては、全照合工程および全照合手段により、部分被検索ビットストリングが取り得る全ての値と照合し、部分被検索ビットストリングに対応する段階の部分登録ビットパターン（以降ではエントリとも称する）を示す現段階のマスクデータから完全一致または範囲一致を導く。マスクデータは、部分エントリのビット列そのものである必要はなく、むしろ、部分エントリの値を直接またはビットパターンあるいはフラグなどにより部分エントリと対応するような形式で示すデータであることがデータ量の点から望ましく、予め部分被検索ビットストリングが取り得る値が決まっているので、そのようなデータでロードすることが可能である。予め部分被検索ビットストリングが取り得る全ての値と照合するので、エントリを追加する際は、基本的にはマスクデータを更新するだけで良く、エントリの追加削除にともなうエントリ移動は発生しない。したがって、上位アプリケーションによるテーブル管理のオーバーヘッドを縮小または排除できる。このため、パケットフィルタリングやステートフルインスペクションにおいても上位アプリケーションソフトウェアの負担が軽いビットストリングの照合方法および照合装置を提供できる。

【0020】

部分被検索ビットストリングが取り得る全ての値と照合する処理は、比較器やルックアップテーブルを用いてハードウェアで高速に実行可能であり、多段階に分割することによりハードウェアの規模も大きくならない。部分被検索ビットス

トリングのビット長は限定されるものではないが、ビット長が長くなればハードウェアは大きくなり、ビット長が短ければ照合段数が増加する。現状では、部分被検索ビットストリングのビット長としては4から5ビット程度が好ましいと考えられる。さらに、検索対象となる被検索ビットストリング（以降ではキーとも称する）のビット長増に対してもハードウェア自体のアルゴリズム的な変更は必要なく、ユニットを追加したり、一連の処理を繰り返すことで対応可能となる。

【0021】

そして、全照合工程とマスクロード工程とを並列に行うことが可能であり、多段階に分割することに伴い発生するメモリからマスクデータをロードするオーバーヘッドの時間も大幅に削減できる。したがって、特殊な構造で高速なメモリを使わなくても、十分な検索速度を発揮でき、廉価で大容量を実現する照合方法および装置を提供できる。そのような照合装置は、収容可能なエントリ数はSDRAM容量にのみ規定されるため、スケーラブル性が高く、また、メモリ管理が柔軟に行えるため、複数種の検索を単一のハードウェアセットで柔軟におこなえるものになることは上述した通りである。

【0022】

さらに、マスクデータをロードする時間のオーバーヘッドを削減するには、メモリに対してデータを入出力するキャッシュ装置を設けることが有効であり、キャッシュメモリの効率を向上し、ミスヒット率を低減するには、メモリ上の記憶単位であり、マスクデータおよびアドレステーブルを含む照合テーブルのサイズとキャッシュラインサイズが同一であることが好ましい。

【0023】

さて、範囲一致検索においては、一致する部分登録ビットパターンがあり、最大または最小の部分登録ビットパターンがあるときには、最大または最小の部分登録ビットパターンに続く次の段階のマスクデータのアドレスを候補アドレスとして記憶あるいは出力しておくことが望ましい。照合の現段階において、一致する部分登録ビットパターンがなく、最大または最小の部分登録ビットパターンもないときには、最大または最小の部分登録ビットパターンが発生する段階まで遡るバックトラックが発生する。そのときに、候補アドレスが記憶されていれば、

出力工程において、候補アドレスを含む照合継続情報を出力することにより、バックトラックにより照合を継続する段階まで最短の処理時間で到達できる。

【0024】

範囲一致検索において、最大または最小の部分登録ビットパターンに続く現段階のマスクデータのアドレスを含む照合継続情報が与えられたときは、現段階のマスクデータに示された最大または最小の部分登録ビットパターンに続く次の段階のマスクデータのアドレスを辿ることにより、被検索ビットストリングよりも値が小さくて最大の登録ビットパターン（最大の登録ビットパターン）、または、被検索ビットストリングよりも値が大きくて最小の登録ビットパターン（最小の登録ビットパターン）に到達することができる。したがって、その状態では、全照合工程および判定工程をバイパスすることができ、出力工程および出力手段では、現段階のマスクデータに対応するアドレステーブルから、現段階のマスクデータに示された最大または最小の部分登録ビットパターンに続く次の段階のマスクデータのアドレスを含む照合継続情報を出力すれば良い。

【0025】

このように段階毎のマスクデータの最大または最小を辿りながら最大または最小の登録ビットパターンにたどり着く処理は、エントリの登録、削除がない限りマスクデータが決まればたどり着く登録ビットパターンは同一になる。したがって、マスクデータと共に、そのマスクデータに示された最大また最小の部分登録ビットパターンに対応して決まる最大または最小の登録ビットパターンを示すバイパスデータをメモリに記憶しておくことが望ましい。最大または最小の部分登録ビットパターンに続く現段階のマスクデータのアドレスを含む照合継続情報が与えられたときは、全照合工程、マスクロード工程および判定工程をバイパスし、出力工程および出力手段において、現段階のマスクデータに対応するバイパスデータを含む最終照合情報を出力して被検索ビットストリングの照合を終了することができる。したがって、中間の段階のマスクデータを辿る処理を省くことができる。

【0026】

特に、上述したバックトラックが発生したときでも、その後の処理は候補アド

レスに示されたマスクデータに対応するバイパスデータを出力するという処理で終了し、ほとんどバックトラックによるペナルティーは発生しない。

【0027】

上述したように、本発明の照合方法および照合装置においては、登録ビットパターンを追加あるいは削除する際に、エントリの移動は発生せず、単に、マスクデータおよびアドレステーブルを更新することにより、登録ビットパターンを追加または削除できる。したがって、照合装置に、そのような更新工程を実行する更新手段を設けておくことにより、更新するためのデータをパケットとして上流から下流に流して、エントリの追加削除を行うことができる。

【0028】

本発明の照合方法および照合装置は、さらに、検索対象となる被検索ビットストリングのビット長増に対してもハードウェア自体のアルゴリズム的な変更は必要なく、ユニットを追加したり、一連の処理を繰り返すことで対応できるので、複数の照合段階あるいは照合装置をパイプライン方式で接続することにより、検索時間の実質的な短縮を実現できる。すなわち、複数の照合段階または照合装置の間では、照合継続情報を含むデータをパケット化して伝達することにより、データフロー型の処理で検索処理を実行できる。

【0029】

複数種類の被検索ビットストリングを対象とする検索を実行する必要がある場合は、照合装置を複数有する分類装置において複数種類の検索を順番に実行しても良い。さらには、複数の分類装置を備えた検索装置を提供して、複数の分類装置により分類処理を並列に実行してもよい。もちろん、複数種類の被検索ビットストリングを連結して1つの被検索ビットストリングを用意し、照合を行うことも可能であるが、下流におけるマスクデータの数が膨大になる可能性があり、実際的でない。照合により得られる最大または最小の登録ビットパターンが複数の分類結果を含む範囲を示している場合は、複数の分類結果の論理積を演算し、最終の分類結果を得る必要がある。この論理演算工程および論理演算手段においては、複数の分類結果の論理積を行列演算し、最終の分類結果を得ることができる。このような論理演算工程および手段により、処理速度を向上でき、構成を簡易

にできる。

【0030】

分類装置において、さらに処理速度を向上するのに最も効果的であると考えられるのは、メモリに対してデータを入出力するキャッシュ装置である。そして、マスクデータおよびアドレステーブルを含む照合テーブルのサイズとキャッシュ装置のキャッシュラインサイズが同一であることが望ましいことは上述した通りである。さらに、分類装置で異なる検索を行えるように、最初の照合テーブルを指定する検索種別情報を受信する手段を有する場合は、キャッシュ装置は、それぞれの検索種別および照合段階の単位でキャッシュラインを割り当て、検索種別および照合段階の単位でキャッシュラインを管理する管理手段を備えていることが望ましい。あるいは、メモリでは、それぞれの検索種別および照合段階の単位で割り当てられた個別アドレス領域に照合テーブルが記憶し、キャッシュ装置のキャッシュラインは、個別アドレス領域単位で割り当てられていることが望ましい。ビットストリングの検索においては、全体が同一または上位が同一のビットストリングが供給されることが多い。このため、従来のCPU用のキャッシュシステムのように、メモリ上のデータの局所性を期待したキャッシュシステムを採用するよりは、検索種別および段階に区分けして、その単位で過去の参照テーブルを保存するキャッシュ装置の方が、キャッシュヒット率が高く、検索あるいは照合処理速度の向上に効果がある。

【0031】

また、キャッシュ装置としては、照合テーブルがオンキャッシュでなければ照合装置にその旨を通知し、その照合装置の処理を中断して待ち行列に戻す、要求応答型のデータキャッシュも有効である。照合テーブルがオンキャッシュでないときに、照合装置、データキャッシュおよびメモリが、そのアクセスに占有されてしまい、処理が遅れることを防止できる。パイプライン的に照合装置が接続されている場合は、1つ遅れると、その遅れは後段に影響するからである。また、被検索ビットストリングに識別情報を付加することにより、入出力の順序が保障される必要はなくなるので、待ち行列で待機させても照合処理には影響がない。被検索ビットストリングに識別情報を付加して照合する方法は、範囲一致検索

において、バックトラックが発生したときにも有効である。

【0032】

また、全体が同一の被検索ビットストリングが供給される頻度も高い。したがって、分類装置の分類結果を記憶した履歴キャッシュ装置であって、被検索ビットストリングを照合装置に供給する前に、当該分類装置の分類結果と被検索ビットストリングとを比較する履歴キャッシュを設けることにより、検索あるいは照合処理速度を向上できる。

【0033】

ビットストリングの検索あるいは照合を高速で処理することができる本発明は、デジタルデータを取り扱うすべての処理で有用である。MPEGなどの画像データを示すビットストリームから特定のビットストリングを検索したり、膨大な文字データから特定の条件を検索するなどの多様なアプリケーションが考えられるが、特に高速性が要求されるコンピュータネットワークを対象とするデータ管理装置の分野に有効である。たとえば、分類装置や検索装置の照合あるいは検索により特定されたルートにパケットを送出する機能を備えた、高速ルーティングが可能なデータ管理装置あるいは管理方法を提供できる。また、ルータやサーバなどで必要となるパケットフィルタリングあるいはSPD検索といった範囲検索を高速で実行できるデータ管理装置を提供できる。さらに、TCPパケットのリアセンブルのように、データブロックの先頭アドレスおよび末尾アドレスが一致するだけでなく、前後および包含関係になることもあるフラグメント化されたデータブロックを再構成するアプリケーションにおいても有効である。

【0034】

さらに、本発明の照合方法および照合装置は、大規模なデータベースから有用なルールあるいはパターンを高速に発見するデータマイニングと呼ばれる分野でも有効であり、応用範囲は非常に広い。

【0035】

【発明の実施の形態】

以下では、本発明に係るビットストリングの検索装置をルーティングテーブル検索に使用したルータ（ルーティング装置）について説明する。図1に示すルー

タ 1 は、ネットワーク上のパケットデータを管理する装置であり、パケット ϕp の入出力を制御および管理するパケット管理機能 3 と、パケット ϕp の転送先を判断するためのルーティング検索機能 4 と、パケット ϕp を転送する条件を判断する条件判断機能 9 とを備えている。ルーティング検索機能 4 は、入力されたパケットの経路を示すエントリを複数備えたルーティングテーブルが記憶されたメモリ (SDRAM) 11 と、検索装置 10 と、検索装置 10 に対して検索種別を設定するなどの機能を備えた制御ユニット 12 とを備えている。ルーティング検索機能 4 では、パケット ϕp に含まれる検索対象のビットストリング (被検索ビットストリング) ϕo 、たとえば、IP ヘッダの受信先アドレス (DIP) をメモリ 11 に記録されたルーティングテーブルの各エントリと一致検索を行う。そして、その結果 ϕr に基づき、パケット管理機能 3 が一致したエントリに設定された経路 (インタフェース) へパケットデータを送出する。

【0036】

条件判断機能 9 は、さらに幾つかの機能に分割されている。たとえば、ネットワーク上のセキュリティーサービスを提供する IP セキュリティー (IPsec) 機能 8 と、ネットワーク上のクオリティーサービスを提供する QoS 機能 7 と、ファイアウォール機能 6 などがある。これらの条件を判断する機能の多くは、パケット ϕp を管理する IP ヘッダのビットストリングを被検索ビットストリング ϕo として、その被検索ビットストリング ϕo が属する範囲を検索することが要求される。IPsec (SPD) 機能 8 では、制御ユニット 8c の制御のもと、検索対象として供給された被検索ビットストリング ϕo と、メモリ 8m に記憶された SPD テーブルとを検索装置 8s により照合する。QoS 機能 7 では、制御ユニット 7c の制御のもと、被検索ビットストリング ϕo と、メモリ 7m に記録されたフロー検索テーブルとを検索装置 7s により照合する。さらに、ファイアウォール機能 6 では、制御ユニット 6c の制御のもと、被検索ビットストリング ϕo と、メモリ 6m に記録されたパケットフィルタリングテーブルとを検索装置 6s により照合する。

【0037】

パケット管理機能 3 は、ネットワークを介してパケット ϕp を受信および/ま

たは送信可能な1つまたは複数のパケット入出力部2 aと、受信したパケット ϕ pをキューイングするメモリ2 bとを備えている。キューイングされたパケット ϕ pは、ルーティング検索機能4でルーティング先が決まり、SPD機能8、QoS機能7およびファイアウォール機能6などにおける処理が決定された後に、それらの機能で決定された適切なタイミングで、パケット管理部3の何れかのパケット入出力部2 aから適切なルートあるいはネクストホップに送出される。また、SPD機能8などの照合結果によってはキューイングされたパケット ϕ pが出力されずに破棄されることもある。

【0038】

本発明の照合方法を採用した検索装置は、これらの各機能4、6、7および8の検索装置10、6 s、7 sおよび8 sとして適用可能であり、IPv6に基づいて被検索ビットストリング ϕ oのビット長が増しても高速でサーチし結果 ϕ rをパケット管理部3に返すことができる。条件判定機能9に含まれる機能6、7および8の検索装置6 s、7 sおよび8 sにおいては、被検索ビットストリング ϕ oが属するルールを見出すために範囲一致検索あるいは照合が実行される。一方、ルーティング検索機能4においては、ルートを特定するために完全一致検索または最長一致検索が実行される。ルーティング検索機能4において、プレフィックス以降の最長一致の結果を得る最長一致検索は、プレフィックス以降のすべてのビットを最小側と最大側に展開した範囲一致検索に還元される。なお、本明細書において、バイナリビットツリーの「0」の方向を最小側(MIN)と呼び、「1」の方向を最大側(MAX)と呼ぶ。

検索装置の基本原理

以降においては、本発明に係る検索装置の概要を説明する。本発明の検索装置は複数種類の検索を単一のハードウェアで行えるものであり、いずれの機能の検索装置としてもほとんど構成を変えずに適用可能である。したがって、以降においては検索装置10を、完全一致検索および範囲一致検索が可能な装置として代表して説明する。

【0039】

検索装置 10 は、バイナリビットツリーを構成するエン트리群と被検索キー（ビットストリング）の間の大小判定アルゴリズムを用い、エン트리（登録ビットパターン）群が表現する範囲（ルール）のどこに被検索キーが属するかを判定するハードウェアである。バイナリビットツリーに基づく照合を実行するアルゴリズムをハードウェアで実現するにあたり、バイナリビットツリーを短いビット長のテーブルに分け、多段階で照合を行う。各照合段階では、各段のテーブル内の大小判定を並列ビット照合により行う。多段階の照合にすることにより、検索対象となるビットストリングのビット長の制限はなくなり、汎用的な利用が見込まれる。また、多段階の照合にすることにより、各段階で比較するビット長を限定でき、予めすべてのエントリが登録されているものとして並列に照合しても処理時間への影響はなく、ハードウェアが増大することも無い。そして、各段に登録されているエントリを示すテーブルを示すデータ（以降ではマスクデータ）は圧縮することが可能であり、バイナリビットツリー作成（登録・削除）コストが軽減され、多段テーブルによる圧縮効果と相まって必要メモリ量も抑えられる。

【0040】

照合過程の各段階における並列ビット照合は逐次実行プロセッサ上のソフトウェアでは実現不可能であり、並列処理ハードウェアあるいは並列実行型のプロセッサで実行することになる。並列ビット照合に必要とされるハードウェアは簡易なものであり、並列実行型のプロセッサを用いずに、経済的、コンパクトに実現することができる。

【0041】

バイナリビットツリーによる同値および大小の判定

以下で述べる範囲検索において、右倒れ型近似ロジック、左倒れ型近似ロジックはハードウェア設計時にどちらかを選んで設計する。どちらを選んでも等価な範囲検索が可能である。両方式を切り替えられるよう設計することも可能である。一度に両方の近似で検索することも可能である。本明細書では、特に記載しない限り、右倒れ型近似ロジックに基づき設計された装置により説明する。

【0042】

図 2 に、バイナリビットツリー上での完全一致検索のロジックを示してある。

ツリーの1段が1ビットに相当するバイナリビットツリーにおいては、上位から1ビットずつ被検索キーとエントリとの間で照合していく。この上位とは照合を開始するビットであり、必ずしも最上位のビット（MSB）でなくても良く、最下位のビット（LSB）であっても良い。バイナリビットツリーのあるノードでの次ノードへの接続は「両方向あり」「0方向のみあり」「1方向のみあり」の3通りしかない。1ノードが1エントリを意味するバイナリツリー（2分木）とは異なり、バイナリビットツリーではノードはエントリの1ビットを意味するに過ぎない。エントリは全て等ビット長なのでツリー上でのノード段数も等しく、上位ノードからの接続が存在するノードを辿っていく途中で下位ノードに対する接続が「両方向なし」となることはありえない。

【0043】

本明細書において、図2（a）および（b）に示すように、「被検索キー21のビットとエントリ22のビットが一致」となるエントリは「照合候補」と定義する。また、図2（c）および（d）に示すように、「被検索キー21のビットとエントリ22のビットとが不一致」となるエントリは「照合失格」と定義する。図2（e）および（f）に示すように、ノード接続が両方向である場合は、被検索キー21のビットとエントリ22aのビットが一致」となるエントリ22aは照合候補で、「被検索キー21のビットとエントリ22bのビットが不一致」となるエントリ22bは照合失格である。

【0044】

完全一致ロジックにおいては、各ビット（各ノード）で照合をおこない、照合候補が存在する間、照合候補の次ノードを辿り、照合を継続する。照合候補が無くなればそこで照合は終了し、検索ミスヒットという最終照合結果が得られる。最終ビット（最終ノード）の照合完了後に照合候補があれば（状態が照合継続であれば）、それに対応するエントリが検索ヒットである。

【0045】

図3は、バイナリビットツリーにおける完全一致検索の照合ロジックを表すものである。バイナリビットツリーに登録されたエントリ群22（値0, 2, 8, 15）に対し、被検索キー21aとして値8を投入し、値8に対応するエントリが

完全一致検索ヒットと判定される例と、被検索キー 21b として値 4 を投入し、検索ミスヒットと判定される例を示してある。最初のビットであるノード A で被検索キー 21a は 1 方向に進む。値 0 および 2 のエントリ 22 は 0 方向、値 8 および 15 のエントリ 22 は 1 方向なのでエントリ 22 は両方向に延び、前者のエントリ 22 は被検索キー 21 のビットと異なるので照合失格、後者のエントリ 22 は被検索キー 21 のビットと一致するので照合継続である。そして、このノード A での照合後に照合候補エントリが存在するため、照合継続である。

【0046】

次のノード E では、被検索キー 21a は 0 方向に進み、値 8 のエントリ 22 が 0 方向、値 15 のエントリ 22 が 1 方向に延びている。したがって、ノード E でも照合候補が存在するので照合継続である。ノード F では、被検索キー 21a は 0 方向に進み、エントリ 22 は片方向であるが、その値 8 のエントリ 22 が被検索キー 21a と同じく 0 方向なので照合候補が存在し、照合継続である。ノード G でも同様であり、最終ノードである。したがって、最終ノード G の照合後に照合候補があるので、完全一致検索は値 8 のエントリ 22 にヒットし終了する。

【0047】

被検索キー 21b では、ノード A では照合候補があるため照合が継続する。ノード B では、被検索キー 21b が 1 方向、値 0 および 2 のエントリ 22 は片方向で 0 方向に延びている。したがって、照合継続となっていた値 0 および 2 のエントリ 22 は照合失格となり、このノードでの照合後に照合候補が残らない。このため、検索ミスヒットとなる。

【0048】

範囲検索でも完全一致検索同様にバイナリビットツリーを上位から 1 ビットずつ被検索キーとエントリとの間で照合していくが、範囲検索の場合は「照合候補」に加え、「近似候補」および「近似失格」の概念がある。完全一致検索での「照合失格」が「近似候補」と「近似失格」に区別されたことになる。また、ロジック上の必要性から「近似仮定」も定義する。

【0049】

図 4 に右倒れ近似の類型を示してある。図 4 (a) および (b) に示すように

、「被検索キー 21 のビットとエントリ 22 のビットが一致」となるエントリは「照合候補」と定義する（これは完全一致検索と同様である）。この照合候補がある間、右倒れ型近似ロジックによる照合は継続する。また、図 4（e）および（f）に示すように、「エントリ 22 が両方向あり」の場合も、「被検索キー 21 のビットとエントリ 22 のビットが一致」である側のエントリ 22 は照合候補となり、照合継続である。

【0050】

一方、図 4（c）および（d）に示すように、「被検索キー 21 のビットとエントリ 22 のビットが不一致」となるエントリは照合候補から外れる。完全一致検索ではいずれの場合も「照合失格」としていたが、右倒れ型近似の場合、図 4（c）のように「被検索キー 21 のビットが 1 でエントリ 22 のビットが 0」となるエントリ 22 を「近似候補」と定義する。逆に、図 4（d）のように「被検索キー 21 のビットが 0 でエントリ 22 のビットが 1」となるエントリ 22 を「近似失格」と定義する。近似ロジックでは、最新の近似候補を「近似仮定」として保持しておく。実際に「近似仮定」として保存が必要な値は、近似候補の指す次ノード位置である。近似候補でも近似失格でも、図 4（c）および（d）に示すようにエントリ 22 が片方向で照合候補がなければ、照合過程は終了する。

【0051】

図 4（e）および（f）に示すように、エントリ 22 が「両方向あり」の場合は、「被検索キー 21 のビットとエントリ 22 のビットが不一致」となる側のエントリ 22 も同様に、定義に従い近似失格あるいは近似候補となる。「両方向あり」なので「被検索キー 21 のビットとエントリ 22 のビットが一致」側のエントリ 22 も存在し、それが照合候補となるので照合は継続する。それと共に、「近似候補」となるエントリ 22、すなわち、図 4（e）の「近似候補」のエントリ 22 が「近似仮定」となり、それまでの「近似仮定」の値は更新される。

【0052】

最終ビットの後に照合継続（照合候補あり）ならば、そのエントリに完全一致したことになる。途中（あるいは最終ビット）で照合終了となった場合、保持してある近似仮定（最新の近似候補）を見て近似ヒットがありえるかどうかを判定

する。近似仮定がなければ検索ミスヒットである。範囲検索の意味的には、定義された全ての範囲の外である。近似仮定があれば近似ヒットである。しかし、近似仮定が途中のノードを指す場合は、そのノードの下に複数のエントリ 22 が存在する可能性があるため、近似仮定のエントリ群 22 から近似決定をおこなわねばならない。この過程は、近似照合とは別のロジックで処理できる。右倒れ近似においては、以下で最大値検索 (MAXサーチ) として説明している処理を行い、被検索キー 21 の値以下の値のエントリの中から、最大の値のエントリ (被検索キーに近い最大のエントリ) を選択して近似エントリとして確定する。

【0053】

図 3 を、バイナリビットツリーにおける右倒れ型近似ロジックによる範囲検索を表すものとする。以下のようになる。値 4 の被検索キー 21 b については、ノード A で 0 方向の値 0 または 2 のエントリ 22 は照合候補、1 方向の値 8 または 15 のエントリ 22 は近似失格となる。ノード A では、照合候補が存在するため照合継続となる。次にノード B では、0 方向の値 0 または 2 のエントリが近似候補となり、近似候補なので近似仮定 (最新の近似候補を保持) の更新を行い、近似候補の指す次ノードであるノード C を近似仮定として保持する。

【0054】

ノード B の照合後に照合候補がなく、照合過程はいったん終了する。近似仮定があるので、近似ヒットである。従って、近似仮定となるエントリ群から MAXサーチにより近似確定をおこなう。この場合、値 0 と値 2 のエントリ 22 から値 2 のエントリを選択することになる。図 3 に示したように、被検索キー 21 b は $2 \leq \text{キー値} \leq 7$ の範囲の値であれば、同じ結果である値 2 のエントリ 22 を得ることができる。つまり、値 2 のエントリ 22 は (2~7) という範囲を意味するエントリとなっている。このため右倒れ近似が範囲検索になるのである。

【0055】

次に、値 8 の被検索キー 21 a においては、ノード A では、0 方向のエントリ 22 (値 0 と 2 が属する) が近似候補となり、近似仮定 (ノード B を指す) が更新される。1 方向がエントリ 22 (値 8 と 15 が属する) が照合候補となり、照合候補も存在するため照合継続となる。ノード E では、ビット値 0 方向のエント

り 22 が照合継続、ビット値 1 方向のエントリが近似失格となる。近似候補はないため、近似仮定は更新されない。ノード F 以降は単方向で照合継続となり、近似候補が発生しないため、近似仮定は更新されない。そして、最終ビット後に照合継続（照合候補あり）なので、値 8 のエントリ 22 に完全一致するという最終照合結果が得られる。

【0056】

さらに、値 14 の被検索キー 21c を投入した場合を考える。ノード A では、ビット値 0 方向のエントリ 22（値 0 と 2 が属する）が近似候補となるので、近似仮定（ノード B を指す）が更新される。また、ビット値 1 方向のエントリ（値 8 と 15）が照合候補となり照合継続となる。ノード E では、ビット値 0 方向の値 8 のエントリ 22 が近似候補となり、近似仮定は更新される（ノード F を指す）。また、1 方向の値 15 のエントリ 22 が照合候補となるので、照合継続となる。ノード H では、単方向で値 15 のエントリ 22 が照合候補となり、近似候補が発生しないので、近似仮定は更新しないで、照合継続となる。ノード I では、単方向で照合失格となる。従って、いったん照合過程は終了する。

【0057】

このとき、近似仮定はノード E での更新でノード F を指している。近似確定のため、ノード F 以下を MAX サーチし、値 8 に対応するエントリ 22 が近似ヒットと決定される。したがって、この被検索キー 21 は、値 8 のエントリ 22 がカバーする範囲（8～14）に属することが判明し、範囲一致検索が実行されたことになる。

【0058】

図 5 に、左倒れ近似における「照合候補」「近似候補」「近似失格」の定義を示してある。右倒れ近似と「近似候補」「近似失格」のビットが入れ替わっただけである。照合ロジックも、上記の入れ替わり以外は右倒れ近似と同様である。ただし、右倒れ型近似では近似確定するため近似仮定に対する MAX サーチをおこなうのに対し、左倒れ近似の場合、近似確定のために近似仮定に対する MIN サーチをおこなう点が異なる。MIN サーチでは、被検索キー 21 の値以上の値のエントリの中から、最小の値のエントリ（被検索キーに近い最小のエントリ）

を選択して近似エントリとして確定する。

【0059】

最終ビットまで「エントリのビットと被検索キーのビットが一致」であった場合、つまり最終ビットの照合後に照合継続であった場合、これも範囲検索としてはヒットとする。つまり、右倒れ型近似による範囲検索とは、完全一致ヒットと近似一致ヒットを区別せず、「キー \geq エントリ」となる最大エントリを検索するものである。ただし、検索の用途によっては「 $=$ 」を含まない、「キー $>$ エントリ」の右倒れ近似とすることもできる。また、「 $=$ 」一致なのか「 $>$ 」一致なのかを区別した結果とすることもできる。

範囲検索におけるエントリデータの形式

右倒れ型近似による範囲検索では、領域は右倒れ近似ロジックに適合するよう正規化されエントリ登録される。この正規化は領域を表す2値（[FROM, TO]）のうちTO（大値）側の補正である。例えば[a, b]で表される領域Rは値aのエントリAと値b+1のエントリBとして登録される。エントリAは領域Rを表現し、エントリBは領域S[b+1, ∞]を表現する。エントリAに右倒れ型近似ヒットすることは、領域Rに範囲検索ヒットすることである。エントリBに近似ヒットすることは、領域Sにヒット、つまり範囲検索ミスヒット（領域R外）を意味する。近似ミスヒットは領域Q[0, a-1]、つまり範囲検索ミスヒット（領域R外）を意味する。

【0060】

一点を表す領域U[c, c]も定義可能であり、値cのエントリCと値c+1のエントリDとして登録される。エントリCは領域Uを表現し、エントリDは領域V[c+1, ∞]を表現する。値cと値c+1の隣接するエントリが登録されるため、値cであるエントリCへのヒットは完全一致ヒットでしかありえず、点領域Uに範囲検索ヒットを意味する。エントリDへの近似ヒットは領域Vにヒット、つまり範囲検索ミスヒット（領域U外）である。近似ミスヒットは領域T[0, c-1]、つまり範囲検索ミスヒット（領域U外）を意味する。

【0061】

左倒れ型近似でも領域の正規化はおこなわれる。ただし、右倒れ型とは逆に F ROM (小値) 側を補正する。

【0062】

エントリと集合の関係を、エントリ I は集合 $SET(I) [i, \infty]$ を意味すると定義する。このとき、領域 $R [a, b]$ は、値 a のエントリ A すなわち集合 $SET(A) [a, \infty]$ と、値 $b+1$ のエントリ B すなわち集合 $SET(B) [b+1, \infty]$ との差集合 $SET(A) - SET(B)$ で表せる。エントリ A に右倒れ近似ヒットするということは、エントリ A にヒットし、エントリ B にミスヒットということである。

【0063】

照合方法

図6に示すように、多ビットのバイナリビットツリー24を多段階または多階層に分けて、さらに各段階を適当な数のエントリが表れるビット幅に分割することにより、テーブル化できる。一定単位の短ビット長の分割テーブルまたは部分テーブル25を規定することにより、多ビットのバイナリビットツリーを、上位の部分テーブル25から指定される各段階の部分テーブル25の組み合わせにより表現できる。そして、固定ビット数の部分テーブルにすることで、並列処理対象エントリは有限固定となり、部分テーブル25に含まれるエントリをハードウェアにより並列照合することができる。さらには、部分テーブル25に含まれる最大エントリ数は有限固定になるので、実際に部分テーブル25に含まれるエントリの有無にかかわらず、被検索キーが部分テーブル25に含まれ得るエントリのいずれと一致するかを、ある部分テーブル25が選択されると自動的にハードウェアで照合することも可能となる。

【0064】

図6では、3ビットの部分テーブル25によりバイナリビットツリー24を表現しており、各部分テーブル25に収容される部分エントリ26は値0～7の8エントリが最大数である。ハードウェア実装としては、メモリのワードのビット長や検索キーのビット長などを考慮し適切なテーブルビット長を決定するが、テーブルビット長が n ビットになると並列処理対象であるテーブル内のエントリ数

が 2^n になるため、一般的には4ビットテーブル程度が適切である。

【0065】

部分テーブル化することにより、各段の部分テーブルでの検索は有限固定エン트리数でのサーチになる。また、被検索キーが固定ビット幅であれば、エントリを示すバイナリビットツリー24は固定長となり、それを表現する部分テーブル25の段数は予め定まった有限固定段数となる。このため、一致検索においてはバックトラックの発生しないバイナリビットツリー検索となるので、検索に要するワースト時間はエン트리数にかかわらず一定時間となる。

【0066】

図7(a)に示すような論理イメージで、バイナリビットツリー24は部分テーブル化されている。図7(b)は、部分テーブル25を汎用テーブル27として構成した様子を示してある。この汎用テーブル27では、1エントリはビットパターン値27aと、それに対応する下位テーブル27へのポインタ27bで構成され、それが最大 n 個あるテーブル構造となる。さらに、図7(c)に、汎用テーブル27を圧縮したテーブル28で部分テーブル25を実現した様子を示してある。この圧縮テーブル28は、エントリの有効無効を示すビットフラグ(マスクデータ)28aと下位へのポインタ28bで1エントリが構成される。下位の圧縮テーブル28へ続くポインタ28bは、圧縮テーブル28が記憶されているRAMのアドレスなどになり、アドレステーブルを生成する。

【0067】

汎用テーブル27と圧縮テーブル28の違いは、汎用テーブル27では登録ごとにエントリにビットパターンをセットしていくのに対し、圧縮テーブル28では、テーブルは固定ビット幅のテーブルであり、ビットパターンを省略している点である。部分テーブル25が固定ビット幅の場合、ビット幅に対応して上限エン트리数が決まる。したがって、例えば4ビットテーブルの場合、エントリNo0はビットパターン0000、エントリNo1はビットパターン0001というようにエントリNoとビットパターンの関係を一対一に固定できる。したがって、アクティブなエントリNoを示すことだけで、部分テーブル25に含まれる部分エントリ26を指定することができる。エントリNoを示すデータはいくつか

考えられるが、図7(c)に示すようにエントリNoに対応したビット位置にオンオフのフラグとなるビットを配置したビットパターン（マスクパターン）28aはハードウェアで利用しやすい形態の情報である。また、マスクパターン28aにより部分エントリの有無を示すことにより、実際の部分エントリのビットパターンを登録する処理および登録するためのハードウェアおよびソフトウェアを省略できる。

【0068】

この部分テーブル25を圧縮テーブル28とした最適化により、部分テーブル25をそれに対応する被検索キーの部分（部分被検索キー）と照合するハードウェアが単純化される。4ビットの部分テーブル25であれば、4ビットの部分エントリ、すなわち、値0から15までの部分エントリが取りうる値と部分被検索キーとの照合を並列に行うハードウェアを採用することにより、照合作業およびハードウェアが単純化される。すなわち、全エントリハードウェア並列照合が可能となり、それが単純化される。また、エントリNoとビットパターンの関係が固定でソート済なので、エントリ（部分エントリ）の追加削除は、基本的にはマスクデータ28aを更新するだけで実現できる。そして、テーブル内でのMIN/MAXサーチはビットパターンの比較なしにエントリNoのみでおこなえる。したがって、範囲一致検索において、近似候補となる部分テーブル25までバックトラックが発生したとしても、その後は、エントリが有効か無効かのフラグ（マスクデータ）28aの参照だけでテーブル内におけるMIN/MAXサーチは終わることができるので、実質的にはバックトラックが発生しない照合方法ということも可能である。

【0069】

MIN/MAXサーチとそのバイパス

MIN/MAXサーチの原理

バイナリビットツリーにおけるMIN/MAXサーチの原理は以下の通りである。図8(a)に、右倒れ近似におけるMAXサーチの概要を示してある。ビット位置（ノード）A1にて、被検索キー21に対してエントリ22がなくなり、右倒れ型範囲検索の照合候補がなくなる。そのため、照合は途中で終了し、近似

仮定エントリに対するMAXサーチを開始する。エントリ22が片方向しかない場合はその方向に辿る。ノードA2においては、0方向にしかエントリ22がないので、それを辿る。次のノードA3においては、エントリ22が0および1の両方向にあるので、大きい方向（ $0 < 1$ なので1の方向）を優先して辿る。各ノードに対して同様の処理を行い、最下段ビット位置A6まで辿り着き、得られるエントリ22がMAXエントリ（被検索キーの値よりも小さくて最大の値をとるエントリであり、被検索キーに近い最大のエントリ）である。

【0070】

図8（b）は、左倒れ近似におけるMINサーチの概要を示してある。左倒れ型範囲検索の場合、同種のロジックで小さな方向（0の方向）を優先して辿り、MINエントリ（被検索キーの値よりも大きくて最小の値をとるエントリであり、被検索キーに近い最小のエントリ）を取得するMINサーチをおこなう。

【0071】

テーブルにおけるMIN/MAXサーチ

バイナリビットツリーをテーブル化すると、MIN/MAXサーチのロジックもテーブル対応にしなければならない。図9（a）に汎用テーブルでのMIN/MAXサーチのうち、右倒れ型近似におけるMAXサーチを示してある。テーブルA1にて照合失格（キー値21と同値のエントリ22が無い状態）となる。テーブル内で「キー値>エントリ値」となるエントリの中から、最大値であるエントリ（「キー値>エントリ値」の中でエントリ値が最もキー値に近いエントリ）を選択し、近似候補エントリ22xとする。近似候補22xの指す次テーブルが近似仮定エントリ22yである。そして、その近似仮定エントリ22y以下に対するMAXサーチを開始する。

【0072】

まず、近似仮定エントリ22yのテーブルであるテーブルA2に移る。テーブルA2内のエントリを調べ、最大値をもつエントリを選択する。そのエントリの次段テーブルに移り、以下0、最大値を持つエントリ選択と下段への移動を繰り返し、最下段まで到達する。最下段でも最大値エントリを選択し、最大値エントリの登録データに辿りつき、MAXエントリを出力する。図9（b）は、左倒れ

型範囲検索の場合であり、同種のロジックでMINサーチをおこなう。

【0073】

図10に、圧縮テーブル28のMIN/MAXサーチのロジックのうち、右倒れ型近似に対応したMAXサーチのプロセスを示してある。まず、圧縮テーブル28ではマスクデータ28aにおいてエントリNoとビットパターンとの関係が固定であり、ソート済の状態に等しいので、テーブル内での最小値/最大値エントリの選択はビットパターンの比較なしに、マスクデータ28aの対応するエントリNoにフラグを立てるだけで行うことができる。つまり、有効無効フラグの参照だけでテーブル内における最小値/最大値エントリの選択は終わる。そして、追加したエントリNoに対応するアドレステーブル28bのレコードにそのエントリNoに続く圧縮テーブル28のアドレスを記録することにより圧縮テーブル28をリストで繋ぐことが可能となり、リニアアドレス上のテーブルでなくてもバイナリビットツリーの多段階検索が可能となる。

【0074】

右倒れ型範囲検索におけるMAXサーチを説明する。テーブルA1のマスクデータ28aにおいて照合候補がなくなり、照合が終了すると、近似仮定エントリに対するMAXサーチを開始する。近似候補の選択は、テーブルA1のマスクデータ28aの中でキー値>エントリ値となるエントリのうち最大値をもつエントリ（最大のエントリ）を選択することでおこなう。ビットマップによる圧縮テーブルなので、エントリNo=エントリ値であり、上記条件に合致するエントリはキー値エントリから左にみて初めて1が立っているエントリが近似候補22xである。この例では照合が終了したのと同じテーブルA1に近似候補エントリ22xが存在する。近似候補エントリ22xの指す次テーブルが近似仮定エントリ22yである。

【0075】

したがって、近似仮定エントリ22yであるテーブルA2に移動する。テーブルA2内で最大値をもつエントリを選択する。エントリNo=エントリ値なので、右端からみて最初に1の立つエントリが最大値エントリ22mである。最大値エントリ22mの指す次段テーブルA3に移り、エントリ選択と下段への移動を

繰り返し、最下段エントリまで到達する。最下段でも同様に最大値エントリ22mを選択し、最下段での最大値エントリ22mの指すデータに到達する。これにより最大のエントリを得ることができる。左倒れ型範囲検索では、同種のロジックでMINサーチを行う。

【0076】

MIN/MAXサーチのバイパス

圧縮テーブル28ではエントリNoとビットパターンとの関係が固定でソート済なので、MIN/MAXサーチはビットパターンの比較なしにエントリNoのみでおこなえる。有効無効フラグの参照だけでテーブル内におけるMIN/MAXサーチは終わる。しかしながら、エントリの登録・削除がない限り、下位の部分テーブル28を辿って到達する最終的な最大のエントリあるいは最小のエントリは不変である。したがって、毎回、MIN/MAXサーチをするのは無駄であり、図11(a)および(b)に示すように、各テーブルにバイパスTAG(BP)を設け、そのテーブル28以下でMIN/MAXサーチした結果、すなわち、最大なエントリあるいは最小なエントリを記録しておくことによりMIN/MAXサーチを省略することが可能となる。これにより、最大のエントリ、あるいは最小のエントリを検索するプロセスは、近似仮定した圧縮テーブル28にアクセスするだけのプロセスに代わり、検索時間は大幅に短縮される。

【0077】

したがって、近似仮定したエントリを近似確定させる際、近似仮定エントリ22yの指すアドレステーブル28bのアドレス（以降においてはTAG（次段テーブルを指すTAG）と称することがある）を使い、次段テーブルのバイパスTAG(BP)により近似確定する最大または最小のエントリを決定することができる。範囲検索の論理として右倒れ型でおこなう場合はバイパスBPには最大のエントリが記録され、左倒れ型でおこなう場合はバイパスBPには最少のエントリが記録される。複数の範囲検索系があった場合、照合のロジック構成としては右倒れ型か左倒れ型かのいずれかに統一する。バイパスBPに両値を記録することにより、テーブルに記録する情報量の増加、照合用のロジックの追加、エントリの登録削除用のロジックの追加などが発生するが、左右両方式を混在させるこ

とも可能である。

【0078】

さらに、バイパスBPを記録することにより、MIN/MAXサーチにおけるバックトラックの遅延も解消することができる。MIN/MAXサーチとは、近似候補エントリ群から近似結果を確定させるための検索である。ところで、右倒れ型であっても左倒れ型であっても、テーブル型ビットツリーの場合、照合継続中に毎段で近似候補が更新されていくわけではない。ある段で近似候補が更新され、その後、数段の近似候補更新を伴わない照合継続があり、下段のほうで照合失敗となった段階で、上段で更新された近似候補を使い、結果を確定するためにさらにMIN/MAXサーチをおこなう必要がある場合がある。

【0079】

図12に一例を示してある。上位のテーブル28から下位のテーブル28に照合継続し、照合失格となった段階で、近似仮定に基づき上位のテーブル28に戻り、別ロジック(MIN/MAXサーチ)で再びツリー状にテーブル28を降りていくことになる。これはバックトラックといえることができる。照合失敗によりMIN/MAXサーチでツリーを辿るロジックが変わるので、MIN/MAXサーチでは再度のバックトラックは発生しないが、照合失敗位置と近似仮定位置が離れている場合、バックトラックと目されるMIN/MAXサーチのコストは重い。

【0080】

これに対し、図13に示すように、バイパスBPを記録しておけば、一度バイパスBPの値を読むことにより、最大または最小のエントリが確定するため、実質的にはサーチプロセスとなるような時間の係る処理は不要となり、実質的にはバックトラックの発生を抑えられる。図13は、右倒れ型の例であり、テーブルA1で近似仮定更新・照合継続となり、テーブルA2では近似仮定更新無し・照合継続となり、テーブルA3で近似仮定更新無し・照合失敗となる。このとき、テーブルA3での照合失敗で近似確定のため、MAXサーチに移行するが、最新の近似仮定はテーブルA2'であるため、テーブルA2'にアクセスする。そして、テーブルA2'にバイパスBPが記録されていれば、それにより直接、最下

段の最大のエントリまで移動でき、照合過程は終了する。

【0081】

検索装置、分類装置およびバイナリビットツリー照合器

図14に検索装置10の概要を示してある。この検索装置10は、各種の検索を行う分類器15を複数備えており、各種の検索を並列に行えるようになっている。もちろん、1つの分類器15により各種の検索をシーケンシャルに行っても良い。検索装置10に対しては上流のアプリケーション、たとえば、図1に示した制御ユニット12、8cなどから被検索ビットストリングであるキー21と、その検索種別を示すルートTAG31と、検索ジョブを識別する情報(ID)32とが供給される。たとえば、SPD検索機能8であれば、1つのパケットデータ ϕp を管理するために、ソースIP、デスティネーションIP、ソースポート、デスティネーションポートおよびプロトコルを示すビットストリングを少なくとも照合して、それらの全てを満足するルールを検索する必要がある。このような場合は、ソースIP、デスティネーションIP、ソースポート、デスティネーションポートおよびプロトコルを示すビットストリングをそれぞれ異なる分類装置15に対して被検索ビットストリング(被検索キー)21として供給し、それぞれのビットストリングと比較するビットパターン(エントリ)22が登録されたバイナリビットツリーの最上位の部分テーブル25(28)を示すメモリ11あるいは8mのアドレスをルートタグ31として供給する。これにより各々の分類装置15においては、供給された被検索キー21を、その被検索キー21の範囲検索に用いるべく予め登録されているビットパターン(エントリ)22と照合することができる。

【0082】

さらに、検索装置10は、複数の分類装置15に供給された被検索キー21の照合結果により得られるルールの論理積を求めて最終結果を取得するための論理演算装置(AND化器)19を備えている。各々の分類装置15における検索処理時間は、各々の分類装置15に供給される被検索キー21のビット長により異なり、さらに、照合過程において上位の部分テーブル28に戻るバックトラックが発生するか否かによっても異なる。したがって、分類装置15から同期して検

索結果が出力されない可能性がある。このため、分類装置 15 に供給される被検索キー 21 に ID 32 を付加することにより、論理演算装置 19 における論理演算を可能とし、さらに、パケット管理装置 3 において管理されるパケットデータ ϕp との関連付けも可能なようにしている。

【0083】

本例の分類装置 15 は、パイプライン状に接続された複数の照合装置 50 を備えており、これらの照合装置 50 の間ではデータパケットとして標準化された照合継続情報（入出力パラメータ）33 が伝達され、パケット駆動される構成となっている。したがって、分類装置 15 は複数の照合装置 50 によりデータフローが形成されており、クロック同期のハードウェアで実現することも可能であるし、データフロー型のハードウェアで実現することも可能である。分類装置 15 は、さらに、圧縮された部分テーブル 28 が記録されたメモリ 11 と照合装置 50 との間の入出力速度を向上するためのバースト転送バッファ 16 も備えている。

【0084】

以下で説明する検索装置 10、分類装置 15 および照合装置 50 の構成は、ソフトウェアで実現することも可能であるし、ソフトウェアを主体として部分的にハードウェアを用いて実現することも可能である。しかしながら、実行速度を考慮するとできるかぎりハードウェアで実現することが望ましい。本発明に係る部分テーブル 25 あるいは 28 を用いた検索方法は、単一、あるいはほぼ単一の構成の照合装置（照合器）50 をパイプライン状に接続することによりビット長の変化に対応でき、また、ルートタグ 31 といった検索種別を示す情報が提供されることにより単一の構成の照合装置 50 により多種類の検索を実行できるようになっている。したがって、本発明の照合方法は、非常にハードウェア化しやすい照合方法であり、経済的で、処理速度の速い検索装置を提供できる。また、上位のソフトウェアあるいはアプリケーションからは、被検索キー 21 と、検索種別情報 31 を提供するだけで、所望の検索を実行できるので、上位のアプリケーションの負荷も非常に小さい。

【0085】

図 15 に、8 個の 4 ビット型の照合器 50 をパイプライン状に繋いで 32 ビッ

トの被検索キー 21 に対する検索を行う分類装置 15 の概略構成を示してある。照合器ハードウェアの構成である。照合継続情報 33 に含まれるルートタグ (RotTAG) 31 および照合タグ (NextTAG) 34 は、SDRAMメモリ 11 の上の圧縮テーブル 28 を指すポインタ (アドレス) である。各照合装置 50 は、照合継続情報 33 に含まれる被検索キー 21 から順番に 4 ビットずつを部分被検索キー 36 として受け取り、ルートタグ 31 または照合タグ 34 により指定されるテーブル 28 からマスクデータ 28a を読み取り、近似一致による比較をおこなう。比較によりヒットしたエントリ (部分エントリ) 26 に対応する次段の圧縮テーブル 28 のポインタ (NextTAG) 34 を、圧縮テーブル 28 のアドレステーブル 28b から読み取り、照合継続情報 33 に含めて次の照合装置 50 に供給する。

【0086】

被検索キー 21 や照合タグ 34 などの照合継続情報 33 は、別々に供給されても良い。しかしながら、データパケットとしてまとめて入力することにより、データフロー化しやすい。また、検索対象となる部分被検索キー 36 は、上位のプログラムにより予め分割された状態で照合装置 50 に供給されても良い。しかしながら、被検索キー 21 の値は全ビットが照合継続情報 33 に含められ、シフトしながら必要なビット数ずつを部分被検索キー 36 として使用していくことが望ましい。これは、照合器パイプライン構成の途中でパイプラインステージを逆流 (バイパス等によるバックトラック) する可能性があり、被検索キー 21 を照合器 50 の外部で切り分けたのでは、パイプラインの進行 (クロック) 同期で適切なステージ (段階) へ投入していくことができないためである。また、SDRAM 11 に対するアクセスの状態 (競合・排他) や、データキャッシュをつけた場合のオンキャッシュかどうか等で各照合装置 50 における処理速度は変動する可能性が高い。したがって、照合器パイプラインの進行自体をクロック同期ではなく要求応答型 (データフロー型) とした方が処理効率は高く、この点でも、被検索キー 21 を照合継続情報 33 に含め、さらに、照合検索情報 33 をデータパケット化して次の段階の照合装置 50 に供給するメリットがある。

【0087】

図 16 には、分類装置 15 の異なる例を示してある。この分類装置 15 は MA

X/MINサーチを実行する際のバイパス用のバス (BPバス) 39を備えており、照合装置50は、BPバス39に出力するインターフェイスを備えている。したがって、この分類装置15においては、各照合装置50が非同期でテーブル28にアクセスする頻度が上昇する。図15に示したように、各段のテーブル28を同一のSDRAM11におくと、パイプラインを構成する各照合器50のメモリアクセス競合があるため、パイプラインが有効に機能しない。従って、各段のテーブル28を個別のメモリ11aにおき、各照合器50のメモリアクセスを独立としたほうがパイプラインの有効性が高い。この照合装置パイプラインを備えた分類装置15では、照合器50の (メモリアクセスを含む) 所要時間をTとし32bitの被検索キー21が途切れなく入力されれば、検索のスループットT (本当は $1/T$)、レイテンシは8Tである。

【0088】

照合器50の構成としてはSDRAM11を除いて処理時間が一定していてパイプラインを組みやすい。データキャッシュ制御によりSDRAMアクセスを排除する履歴キャッシュ方式や専用SRAM型履歴キャッシュ方式を使えばアクセス時間を一定化でき、パイプラインを構成しやすい。これに対し、共有型データキャッシュであれば、キャッシュのスループットと全パイプラインステージのスループットの合計が等しい、あるいは共有型データキャッシュの方がスループットを高くする必要がある。

【0089】

図17に、分類装置15のさらに異なる例をしめしてある。上記の例では、複数の照合器50を処理パイプラインとして並べた構成であるが、省リソースを優先した場合は、図17に示したように、1個の照合器50を繰り返し使う構成も可能である。この場合、照合器50からのメモリアクセスはある時点では常に1アクセスとなるので、テーブル28を全て同一のSDRAM11に置くことで処理速度の優劣はない。

【0090】

照合装置の構成

図18に照合装置50の構成を示してある。この照合装置50は、被検索ビッ

トストリング（被検索キー）21を複数ビットの部分被検索ビットストリング（部分被検索キー）36に分けて、予め登録された複数の登録ビットパターン（エントリ）22と多段階で照合するための1つの照合段階を実行する照合装置である。照合装置50は、照合継続情報33に含まれるコマンド37をデコードして照合装置50における処理を制御するコマンドデコードユニット51と、被検索キー21から、現段階の部分被検索キー36を選択し、その部分被検索キー36が取り得る全ての値と照合する比較ユニット（全照合手段）52と、メモリ11からその照合段階の圧縮テーブル28のマスクデータ28aをロードするロードユニット53とを備えている。ロードユニット53は、圧縮テーブル28をメモリ11からバッファ59に格納するテーブルロード部53aと、バッファ59に格納された圧縮テーブル28からマスクデータ28aを読み出すマスク読み出し部53bとを備えている。圧縮テーブル28にマスクデータ28aと共に格納されているアドレステーブル28bおよびバイパスデータBPは多くの処理で必要になる。このため、本例では圧縮テーブル28をバッファ59にロードしている。マスクデータ28aは、上述したように、エントリ22の部分ビットパターン26であって、先行する段階から得られる照合継続情報33により決まる、部分被検索キー36に対応する段階の部分登録ビットパターン（部分エントリ）26をフラグ（ビットパターン）により示すものである。

【0091】

照合装置50は、さらに、比較ユニット51における全照合結果とマスクデータ28aに、部分被検索キー36に一致する部分エントリ26の有無、すなわち一致を示すことができる判定ユニット54と、判定ユニット54の照合結果により、アドレステーブル28bから照合により特定された部分エントリ26に続く次の段階のマスクデータ28aのアドレス（タグ（本例では圧縮テーブル28のアドレス））を照合タグ（照合候補）34として出力する出力ユニット55とを備えている。この照合装置50は、一致検索のみではなく範囲検索も行えるように、判定ユニット54は、部分被検索キー36に値が最も近い最大（右倒れ型近似について記載し、左倒れ型近似については、最大を最小に変更することによりほぼ対応できる）の部分エントリ26の有無を示すことが可能である。本例の判

定ユニット 54 は、比較ユニット 52 の出力をマスクデータ 28a によりマスク処理するマスク器 (VLD) 54a と、マスク器 54a の出力から状態を選択する選択器 54b とを備えている。

【0092】

出力ユニット 55 は、最大の部分エントリ 26 に続く次の段階のマスクデータ 28a のアドレス (圧縮テーブルのアドレス) を照合タグ 34 として出力する。さらに、出力ユニット 55 は、判定ユニット 53 において、最大の部分エントリ 26 が見つかったときは、その最大の部分エントリ 26 に続く次の段階のマスクデータ 28a のアドレス (圧縮テーブル 28 のアドレス) を候補アドレス (候補タグ、近似候補) 35 として出力し、照合継続情報 33 のパケットデータに保存する。したがって、一致する部分エントリ 26 があり照合継続の場合は、最大の部分エントリ 26 が見つかる と候補タグ 35 にそれが保存され、一致する部分エントリ 26 がない照合終了で、さらに、その照合段階では最大の部分エントリ 26 がないときに、候補タグ 35 を照合タグ 34 として照合継続情報 33 に出力することにより MAXサーチに移行することができる。

【0093】

さらに、照合装置 50 の出力ユニット 54 は、MAXサーチに移行したときに、圧縮テーブル 28 からバイパスデータ BP を読み取って最終照合結果として出力することができる機能も備えている。したがって、出力ユニット 55 は、照合状態などを反映したコマンド 37 を生成するコマンド更新部 55a と、照合タグ 34 を更新する照合タグ更新部 55b と、候補タグ 35 を更新する候補タグ更新部 55c と、バイパス読み出し出力部 55d とを備えている。そして、出力インターフェイス 56 からは、出力ユニット 55 により更新されるパラメータ 34、35 および 37 と、入力された照合継続情報 33 に含まれる ID 32 と、被検索キー 21 とを含むデータパケットが照合継続情報 33 として下流の照合装置 50 に供給される。

【0094】

圧縮テーブル 28 にバイパスデータ BP が設定されていないときは、照合終了を示すコマンド 37 を受け取ると、出力ユニット 55 の照合タグ更新部 55b が

マスクデータ28の最大の部分エントリ26に続く次の段階のマスクデータのアドレスを照合タグ34として出力し、MAXサーチが継続される照合装置50のパイプラインを制御する。

【0095】

さらに、本例の照合装置50は、マスクデータ28aおよびアドレステーブル28bを更新することにより、エントリを追加または削除する更新ユニット60と、更新されたテーブル28をメモリ11に書き込むテーブルライトユニット61とを備えている。更新ユニット60の動作については、以下でさらに詳しく説明する。

【0096】

図19に、照合装置50における処理をフローチャートにより示してある。まず、ステップ101でコマンド37がエントリの追加削除を示しているときは、ステップ102で更新ユニット60によりテーブル28を更新する。ステップ103でコマンド37から前段階の照合が近似で照合終了していることが判明すると、以下に説明する照合過程のステップ104、105および106をバイパスし、出力段階110のステップ111でバイパス読み出し出力部55dによりバイパスデータBPを図16に示したバイパスバス39に出力して照合を終了する。

【0097】

コマンド37より照合を継続する場合は、ステップ104において比較ユニット52により部分被検索キー36を比較し、ステップ105においてロードユニット53によりマスクデータ28aをロードし、ステップ106において判定ユニット54により照合結果を得る。そして、ステップ107において一致する部分エントリ26があれば照合継続なので、出力段階110のステップ109において候補タグ更新部55cにより候補タグ35を更新し、ステップ114で照合タグ更新部55bにより照合タグ34を更新し、コマンド37をコマンド更新部55aにより照合継続にセットして出力する。

【0098】

一方、一致する部分エントリ26がなく、照合終了であるが、ステップ108

で近似の部分エントリ（最大の部分エントリ）26がある場合は、出力段階110のステップ113において照合タグ更新部55bが近似の部分エントリ26に続くタグアドレスを照合タグ34にセットして出力する。ステップ108で近似の部分エントリ26がない場合は、出力段階110のステップ112において、照合タグ更新部55bが、上流の照合過程で得られ、照合継続情報33に含まれて伝達された候補タグ35を照合タグ34にセットしてバイパスバス39に出力する。これにより、MAXサーチが開始される。

【0099】

これらの工程は、実際には、独立して処理できる工程は並列に実行される。たとえば、コマンドを解釈する工程101および102、部分被検索キー36を比較する工程104と、マスクデータをロードする工程105とは独立した処理であるので、並列に実行できる。テーブルをロードするフェーズはSDRAMアクセスであるため時間がかかり、比較フェーズなどとは並列に実行することによりメモリアクセスのオーバーヘッドを緩和できる。したがって、読み出し速度が非常に速い特殊なメモリを用いなくても本例の照合装置50においては十分な検索スピードを確保できる。また、出力段階110において異なるパラメータを更新するフェーズも相互独立であり、コマンド更新部55a、照合タグ更新部55bおよび候補タグ更新部55cも独立しており、これらにおける処理は並列に実行可能である。したがって、図20に示すように、通常の照合モードの場合、照合装置50においては、テーブルロード53と比較52などが並列で実行される。

【0100】

本例のロードユニット53は、照合タグ34の指すSDRAMメモリ11上の圧縮テーブル28を照合器50の内部のテーブルバッファ59にロードする。圧縮テーブル28のデータは、必ずしも全て利用されるものではないが、少なくともマスクデータ28aと、アドレステーブル28bの1個あるいは2個のアドレス（タグ）の値は参照される。したがって、各照合器50において共有資源であるSDRAM11に対し、2～3回のリードをおこなうよりは、バーストリードにより1回だけSDRAM11からリードし、照合器内部で必要値を読み出すほうが効率的である。上述したように、SDRAMからバーストリードするには時

間がかかるため、このテーブルロードフェーズからマスクデータ読み出しフェーズは、比較フェーズと並列に実行される。詳しくは後述するが、SDRAM11との間にデータキャッシュを置いた場合、照合器50に内蔵のテーブルバッファは不要であるが、その場合でも、共有資源たるデータキャッシュへのアクセスを複数回おこなうよりも、やはりローカルにコピーしてアクセスするほうが効率的な場合がありえる。

【0101】

図21に、比較ユニット52をさらに詳しく示してある。本発明の照合装置50においては、部分被検索キー36のビット数が固定なので1テーブルのエントリ数は固定であり、また、圧縮テーブルなのでエントリとビットパターンの対応も固定である。したがって、部分被検索キー36の値と各エントリとの比較は固定論理回路により構成可能である。本例の照合装置50は4ビットの比較を行う。このため、16個の比較器CMP0からCMP15を備えており、比較器CMP0（エントリNo. 0）は入力被検索キー36と値0との比較、比較器CMP1（エントリNo. 1）は入力被検索キー36と値1との比較、そして、比較器CMP15（エントリNo. 15）は入力被検索キー36と値15との比較というように各比較器CMP*i*で、入力値*x*と固有値*i*との比較をおこなう。つまり、比較器CMP*i*（*i*は0～15）はそれぞれのエントリに固有の論理回路である。

【0102】

比較ユニット52は、比較器の代わりにルックアップテーブルLUTを用いたハードウェアなどの他のハードウェアで構成することも可能であり、本例に限定されるものではない。

【0103】

図22に示すように、各比較器CMP*i*は入力値*x*として被検索キー4ビットを取り、CMP*i*固有値*i*との比較をおこない、*x*と*i*との比較結果「<」「=」「>」の3値（2ビット）の比較結果（CMP*i*出力）を出力する。すなわち、各比較器CMP*i*は前項のような論理に基づき結果出力するよう比較器毎に論理圧縮し最適化した回路で構成される。

【0104】

図23は、比較ユニット52の異なる例を示している。並列に実行されるテーブルロードフェーズ～マスク読み出しフェーズに時間がかかることを考えれば、必ずしも上記のように16個の比較器CMP_iを並列する必要はない場合がある。図23は、1個の比較器CMP_xを16回繰り返し利用する構成であり、処理時間のバランスが取れるのであればハードウェアを削減できる。もちろん、テーブルロードフェーズ～マスク読み出しフェーズの所要時間との兼ね合いで比較器CMP_iの必要個数は1個に限定されることはなく、2個以上であっても良い。

【0105】

図24に、判定ユニット54のマスク器54aの構成を示してある。マスクフェーズではマスク器VLD54aにより無効エントリのマスク処理がおこなわれる。各エントリをマスクするマスクエレメント54eは各エントリに対し同一論理の回路である。マスクデータ28aは、実際のエントリ（部分エントリ）26の有効無効を各1ビットのマスク値で表現したものであり、比較フェーズでの各エントリ（仮想的なエントリ）の比較結果から、マスクデータ28aにより無効エントリの比較結果を取り除き、実際に登録されている各エントリ26との比較結果に変換する。

【0106】

各マスクエレメント54eは、図25に示すように、比較フェーズのCMP_i出力2ビットとマスク値1ビットの合計3ビットを入力とし、「×」「<」「=」「>」の4値（2ビット）を出力とする。ここで、「×」は無効エントリ、「<」「=」「>」は有効エントリである。

【0107】

図26から28に、判定ユニット54の選択回路54bの選択論理を示してある。マスクフェーズ後の各エントリのVLD出力は、「×」（無効）、「<」（キー36<エントリ26）、「=」（キー36=エントリ26）、「>」（キー36>エントリ26）の4状態であり、この選択フェーズでは、これらのVLD出力の中から完全一致検索、範囲一致検索の照合結果を得て、次段へ繋がるエントリを選択する。図26に示した選択論理は、完全一致検索の選択論理であり、

マスク器VLD出力が「01」となるエントリを選択する。図26に示した選択論理により、VLD出力が「01」のエントリのみ出力 $G_f(i)$ （完全一致を示す出力）が「1」となる。したがって、出力 $G_f(i)$ が1となったエントリ N_o を取得することにより、完全一致のエントリ N_f を得ることができる。また、出力 S_f は出力 $G_f(i)$ が1となるエントリがあったかどうかを意味する。バイナリビットツリー方式ではエントリとしての重複登録はありえない。従って、 $G_f(i)$ の16出力のうち、完全一致エントリがあれば、唯一であることは保証される。出力 $G_f(i)$ による「=」の選択は範囲検索においても照合継続処理で使用する。

【0108】

図27に右倒れ一致検索型の範囲検索の照合継続選択論理を示し、図28に左倒れ一致検索型の範囲検索の照合継続選択論理を示してある。範囲検索の場合、照合継続は完全一致同様の「キー36＝エントリ26」の選択であり、照合終了の選択は、範囲検索を右倒れ一致型でおこなうのであれば「キー36＞エントリ26」となる最大エントリを選択する。左倒れ一致型であれば、「キー36＜エントリ26」となる最小エントリを選択する。したがって、右倒れ一致の場合はVLD出力が「11」のエントリのみ中間出力 $F_r(i)$ が「1」となり、左倒れ一致の場合はVLD出力が「10」のエントリのみ出力 $F_l(i)$ が「1」となる。

【0109】

そして、範囲検索において、近似候補を示す出力 $G_r(i)$ では、出力 $F_r(i)$ が「1」のエントリのうち、最大値のエントリ（エントリ番号が最大のエントリ）のみが「1」となり、その近似候補のエントリ N_o を示す出力 N_r が決定する。出力 S_r は出力 $G_r(i)$ が「1」となるエントリの有無を意味する。同様に左倒れ一致型の場合は、出力 $G_l(i)$ では、出力 $F_l(i)$ が「1」のエントリのうち、最小値のエントリ（エントリ番号が最小のエントリ）のみが「1」となり、近似候補となるエントリ N_o を示す出力 N_l が決定する。出力 S_l は出力 $G_l(i)$ が「1」となるエントリの有無を意味する。

【0110】

図29に、右倒れ一致型の場合の出力の変化を示してある。図29(a)は比較フェーズ後のCMPi出力であり、図29(b)は、テーブルより読み出したマスクデータ28aである。図29(c)はVLD出力を示し、ビット毎の(a)×(b)となる。図29(d)はキー値>エントリ値となるエントリ群の抽出Fr出力であり、図29(e)の中間出力Hrは、出力Frから右エッジ(キー値>エントリ値となる最大エントリ)を抽出したものである。図29(f)は、最終出力Grであり、出力Hrを利用し、キー値>エントリ値となる最大エントリのみとする。これを見てもGr(i)が「1」となるエントリ、すなわち近似候補となるエントリは1箇所であることは明らかである。一方、図29(g)は一致検索の出力Gfを示しており、VLD出力をもとにキー値=エントリ値(01)となる一致エントリを抽出している。本発明の照合方法においては、テーブル28への重複登録がない構造なので、この一致エントリの唯一性は自明である。出力GrでもGfでも、演算の結果、該当エントリは存在しないか唯一存在のいずれかとなる。

【0111】

図30に、左倒れ一致型の場合の出力変化を示してある。右倒れ一致型の場合と同様にGlの16出力のうち、Gl(i)が「1」となる近似候補のエントリは唯一であることは保証される。

【0112】

出力ユニット55においては、判定ユニット54から得られる上記の照合結果に基づいて各パラメータを更新する。照合タグ更新部54bにおいては、選択回路54bにより得られるエントリNo(出力Nf)に従い照合タグ34を更新する。選択回路54bより得られる出力Sfを見て、出力Sfが0であれば更新はおこなわない。照合継続となるエントリがなく候補が確定するからである。出力Sfが1であれば、照合器50にロード済みの現段階の圧縮テーブル28のアドレステーブル28bから、エントリNfに対応するタグ情報(アドレス)を読み出し、照合タグ34を更新する。圧縮テーブル28においては、ワード位置+0がマスクデータ28aであり、ワード位置+1からタグ情報28bが記録されている。

【0113】

候補タグ更新部55cにおいては、選択回路54bにより得られるエントリNo（出力 N_r ）に従い候補タグ35を更新する。候補更新情報である出力 S_r を見て、出力 S_r が0であれば更新はおこなわない。新規候補となるエントリがないからである。候補更新情報 S_r が1であれば、照合器50にロード済みの現段階の圧縮テーブルから、エントリ N_r に対応するタグ情報を読み出し、候補タグ35を更新する。ここで候補タグ情報35には、タグ情報だけでなく、照合段階を示す候補段階情報も記録することが有効である。この候補段階情報は、候補タグに対応するテーブル段位置に設定する。つまり、候補タグがどのテーブル段であるかを示す情報である。この候補段階情報は、パイプラインが照合段数でフィックスされている分類装置15においては、候補確定後のバイパス処理で必要となるものである。

【0114】

タグアドレスからテーブル段位置が一意に決定できるようにテーブル28が配置されている場合は、候補段階情報は候補タグ情報に含まれるため、別処理は不要である。そうではない場合、照合器50は自分がどのテーブル段数の処理を担当するのかの情報を保持し、その情報に従い、次段の情報を候補段階情報として候補タグ情報35の中にセットする。照合器50をパイプライン利用する場合、テーブル段位置とはつまり照合器IDであるので、自器IDが分かり、自器に対しパイプライン次段にあたる照合器IDが分かればよい。パイプライン次段の照合器との関係は固定なので、照合器の初期化時に定めればよい。次段の照合器IDを使用するよりは、テーブルのタグアドレスに照合段階IDを含めてしまうほうが簡素である。データキャッシュを用い、データキャッシュのコントロールをおこなう場合、テーブルは段位置別にグループ化されたアドレスとなるので、それを流用できる。

【0115】

コマンド更新部55aは、選択回路54bにより得られる照合継続情報である出力 S_f に従い、次段に供給するコマンド37を更新する。照合継続情報である出力 S_f が0であれば、照合継続がなくなり候補が確定したことを意味する

ので、「CMD=MATCHRDBP」を出力する。これにより、次段の照合装置 50 では候補タグ 35 と一致している照合タグ情報 34 によるバイパス処理が行われ最終結果が出力される。照合継続情報 S_f が 1 であれば、照合継続であるので、「CMD=MATCHEXEC」を出力する。これにより、次段の照合器装置でも照合タグ 34 により継続して照合が行われる。

【0116】

出力ユニット 55 は、さらに、コマンド更新部 55a により決定した次のコマンド 37 の値に従い、出力先を決定する。次コマンド 37 が「CMD=MATCHEXEC」であれば、照合は継続するので次段の照合装置 55 にパケット化された照合継続情報 33 を出力する。次コマンド 37 が「CMD=MATCHFIN」であれば、バイパス読み出し後の出力処理であるから、出力をスルーとして最終段小照合器の次（出力器）に送り出す。スルー出力が照合器パイプラインを順次通過しながら出力されるのであれば、次の照合器 50 に最終出力を送り出す。バス 39 を使ってショートカットするのであれば、バス 39 に最終出力を出力する。

【0117】

次コマンド 37 が「CMD=MATCHRDBP」であれば、照合は終了し候補が確定するので、候補タグ情報 35 によるバイパス処理をおこない、検索結果を決定する。この際、候補タグ情報 35 は必ずしも次段テーブルを指しているわけではなく、それ以前の段を指している場合がありうる。したがって、次段の照合装置 50 に出力できるとは限らず、候補段階情報により候補タグを扱える段階の照合装置 50 を特定し、照合継続情報 33 を出力する。

【0118】

図 31 にコマンド 37 の一覧を示してある。なお、これは完全一致検索、1 次元範囲検索に加え、後述する登録・削除も含めたものである。

バイパス処理とショートカット処理

図 32 に、図 16 に示した分類装置 15 においてバックトラックのあるバイパス処理を実行している様子を示してある。この分類装置 15 においては、バイパス用のバス 39 を備えており、バス 39 に照合継続情報 33 を出力することによ

りバックトラックを発生させている。図16に示した分類装置15はメモリアクセスタイムを改善するためにメモリ分離型にしているために、あるタグアドレスの指すテーブル28は、それに対応する段階の照合装置50からしかアクセスできない。ところが、バイパス処理移行（候補確定）時に得られる候補タグ情報35は、照合器パイプライン上の次段とは無関係に、自器より前のいずれかの照合器50に属するものである。従って、バイパスバス39により候補タグ35に従って処理パイプラインを逆行できるようにしている。

【0119】

図33に、図15に示したメモリ共用型の照合器パイプラインを備えた分類装置15においてバイパス処理を実行する様子を示してある。メモリ共用型の照合器パイプラインを備えた分類装置15の場合、全ての照合器50がメモリ11に格納された任意のテーブル28にアクセスできるので、バックトラック用のバイパスバスは必要ない。メモリ共用型の照合器パイプラインでは、次段の照合器50がバイパス処理をおこなう。したがって、最終段での次段のバイパス処理発生に対応するため、最終段の照合器50の次にバイパス専用のバイパス処理器58を設ける。照合器Fで照合失敗、すなわち、照合器A～Fのうち最後に更新された候補が確定することにより次段で候補タグバイパス処理が発生した場合、次の照合器Gがバイパス処理をおこなう。照合器Gは本来、ある段のテーブル28の処理パイプラインを構成しているが、メモリ共用型のため、任意の段のテーブル28にアクセスできる。従って、照合器Fで照合失敗し、候補確定が発生し、照合器Gにてバイパス処理をおこなう場合、テーブルB～Gのどの位置の候補であっても、そのテーブルにアクセス可能である。

【0120】

バイパス処理が完了し結果タグ情報（最終照合結果）が得られて以降の照合器50は、結果タグ情報のスルーのみを行う。最終段の照合器Gでは、照合継続が検索成功（完全一致エントリ）を意味し、照合失敗がテーブルB～Gあるいは結果タグのいずれかを指す候補タグが得られる事を意味する。照合器Gで照合継続により検索成功の場合、照合器50から出力されるタグ情報は結果タグ情報であり、それを受けた最終段バイパス処理器58では、検索終了としてステータスだ

けを更新し、バイパス処理は不要である。

【0121】

一方、照合器Gで照合失敗により候補確定の場合、候補タグ情報35がテーブルB～Gのいずれかを指すタグ情報である可能性がある。したがって、バイパス処理器58はバイパス処理を行い、候補タグ情報35の指すエントリからテーブル28にアクセスしてバイパス値(BP)を読み取り、検索終了ステータスにする。照合器Gで候補確定であり、候補タグ情報が照合器Gで更新された場合は、テーブルGに記されたタグが結果タグ情報となるため、更新された候補タグ情報は結果タグ情報、すなわち最終照合結果となっている。従って、バイパス処理器58ではステータス＝検索完了にするだけで、バイパス処理は不要である。

【0122】

以上まとめると、バイパス処理器58は入力の結果タグ情報であればスルーし、そうでない場合は、テーブルB～Gのいずれかを指すタグ情報が得られるので、バイパス処理をおこなう。結果タグ情報か否かの識別は、タグアドレスの上位ビットを識別用の情報として使うか、パイプラインを流れるデータに情報を付与することで行うことができる。

【0123】

図17に示した繰り返し型の分類装置15の場合は、全ての照合器が任意のテーブル28にアクセスできるので上記と同様にバイパス用のバスは必要ない。さらに、繰り返し型の場合、制御用のカウンタ値をバイパス処理を意味する値にし、繰り返し処理の一環として自器にてバイパス処理をおこなうことができるので、バイパス処理器58も不要である。

【0124】

ところで、バイパス処理にて得られるタグ情報は最終結果であり、バイパス処理以降の照合器パイプラインはスルーしてパイプライン最終段まで進み分類装置15の出力となる。また、完全一致検索での照合不一致(検索失敗)あるいは範囲検索での照合不一致または候補なし(検索失敗)の場合も同じく、検索失敗のステータスを持って自器以降の照合器パイプラインをスルーしていく。バイパス処理のためのバックトラックがなければ、分類装置15において、被検索キー2

1を入力した順番と、その照合結果が出力される順番とが保障されるが、バイパス処理のためのバットラックを許容するので、入力順番と出力の順番が同一であることは保障されない。このため、被検索キー 21 に ID 32 をセットにして供給するようにしている。したがって、スルーする場合も、図 34 に示すように、照合器パイプラインをスルーで進んでいく必要はなく、図 35 に示すように、バイパス用バス 39 を転用して照合器パイプラインをショートカットすることが可能となる。

【0125】

さらには、ショートカット先は常に最終出力部なので、図 36 に示すように、バイパス用バス 39 とは別にショートカット用のバス 38 を設けてもよい。バイパス発生とはほぼ同数ショートカットは発生するため、常に決まった宛先のためにバイパス用バス負荷を高める必要はない。バイパス用バス 39 のバイパス処理での利用率が 50% 以下であるなら、ショートカットもバイパス用バス 39 を使う方がハードウェアが増大しないメリットがある。

【0126】

メモリ共用型照合器パイプラインを備えた分類装置 15 においては、バイパス用バスは不要である。このため、図 37 に示すように、ショートカット用のバス 38 を新規に設けることが望ましい。繰り返し型の分類装置の場合は、制御用カウンタ値の判定でバイパス処理後は繰り返し終了としておけばよく、ショートカット経路は不要である。

【0127】

検索テーブルの管理（更新処理）

以下では、照合装置 50 のテーブル更新ユニット 60 においてエントリ 22 を追加あるいは削除する処理について説明する。まず、上述したように、圧縮テーブル 28 はマスクデータ 28a とアドレステーブル 28b で構成される。したがって、マスクデータ 28a にビットあるいはフラグで示された部分エントリを追加したり削除したりするだけで基本的には検索テーブル 28 の更新が可能であり、ソーティングや正規化などの複雑な処理は必要ない。このため、検索テーブルの管理は非常に簡略化され、高速に処理される。したがって、エントリの登録や

削除といったテーブル 28 の管理を、被検索キー 21 の照合処理と並列実行、つまり削除(登録)と検索がパイプライン並列に実行させることが可能となる。ただし、エントリを追加することにより検索テーブルに用意されるバイパス値 BP を更新する必要があるケースがあり、その判断に若干の処理が必要になる。新しい検索テーブルを生成するための処理も必要になる。しかしながら、バイパス値 BP は、検索テーブル 28 にいったん設定すると、照合処理において、近似候補が決定された後の処理をショートカットできるので、検索テーブルの更新時にバイパス値 BP を設定しておくことには意味がある。

【0128】

まず、テーブル 28 の管理と検索がパイプライン並列に実行される状況を想定すると、照合中の整合性を保つための排他管理が要求される。検索(照合)においても、処理対象のテーブル 28 を占有すれば整合性保持は容易になるが、ツリー上位テーブルにおけるシリアル化がボトルネックとなり、照合処理の並列実行を阻害しかねない。したがって、検索においては「排他は確認するが、テーブルは占有しない」アクセスにより検索の間の並列性を生かしつつ、登録・削除側、すなわちテーブル更新処理では「排他を確認し、さらに、テーブルを占有する」アクセスで整合性も維持する。なお、検索と登録(削除)の間では、先行する登録(削除)の占有に対し後続の検索が排他確認の待ちをおこなうことでシリアル化される。

【0129】

このため、検索は上記のように「排他は確認するがテーブルは占有しない」状態で並列に実行される。このため、登録・削除との関係では、先行する登録・削除に対しては排他管理により追い越しが発生しないようにする。後続する登録・削除については、検索の上位構造における並列性起因の問題であり、下位構造たる検索で対応できるものではないため対処しない。

【0130】

登録は「排他を確認し、テーブルを占有する」状態でシリアル化されテーブル単位で逐次実行される。また、各テーブルのバイパス情報を更新する必要があり、その情報は下位から上位へと伝播するものであるため、登録時に何らかの形で

下位から上位への伝播をおこなわねばならない。考えられる方式としては、上位で待機し下位からフィードバックする待ち合わせ方式と、上位から下位に下っていき双方向リンクにより下位から上位に戻る方式との2種ある。待ち合わせ方式の場合、何らかの待ち合わせ機構をハードウェア資源で実現しなければならず、その資源量により並列数やテーブル段数等が制約を受けるが、処理速度的には有利である。ハードウェアによる待ち合わせ機構ではなくメモリによる待ち合わせにすることも可能であるが、メモリアクセスが増えるため、処理速度面での有利性がなくなる可能性がある。双方向リンク方式の場合、資源量起因の制約はないものの、1テーブルのメモリ量増加やリンク読み出しによるメモリアクセス増加（処理速度に影響）など、メモリ周りに不利な点をかかえる。

【0131】

待ち合わせ方式による登録

図38に、待ち合わせ方式による登録の概要を示してある。この方式では、テーブル化されたバイナリビットツリーを上から下に辿りながらバイパスBPの更新が確定したテーブル28は待ちなし（Wなし）で更新処理をおこない、バイパスBPが未確定のテーブル28は各段で待たせていく（Wあり）。新規のテーブル28nを作成する必要がある段階に到達した場合は、新規のテーブル28nを作成した後に、その段階以下の新規なテーブル28nとの連結と、上位で待機中（Wあり）（バイパスBPの確定待ち）となっているテーブル28へのフィードバックを並列におこなう。

【0132】

図38において、テーブルAでは追加するエントリ22の部分エントリ26は既存（エントリが既存）であり、追加（登録）されるエントリの部分エントリ26がMAXエントリ（最大のエントリ）ではない。このため、下位のテーブルに対するエントリの登録がどうなっても、このテーブルAでのバイパスBPの値に影響を与えずバイパス値BPを更新する必要はない。従って、待ち（W）なしでBP更新も不要である。テーブルBも追加するエントリ22の部分エントリ26は既存（エントリが既存）であるが、その部分エントリ26はテーブルBのMAXエントリに該当する。従って、下位での結果次第でバイパス値BPを更新する

必要がありえる。このため、下位が確定するまでは、このテーブルBのバイパス値BPの更新が必要かどうか確定しないため、下位の更新待ちで待機する（Wあり）。

【0133】

テーブルCは追加するエントリ22の部分エントリ26が新規（エントリが新規）であり、その新規エントリ26がMAXエントリとなる。したがって、このテーブルではバイパス値BPの更新が必要になる。この段階で、バイパス値BPの更新が発生したので、上位のテーブル28へ通知する。Wあり状態となっているテーブルBでは下位からの通知を受けてバイパス値BPを更新し、更に上位の待ちがあればそこへ通知する。この例では、テーブルAはWなしなので上位への通知は不要である。

【0134】

一方、テーブルCから下への流れは上記の上位フィードバックと並列に進む。テーブルCで次段が新規テーブル28nであると判明するので、空テーブル管理キュー69に要求し、応答としてテーブルDになるべきテーブルのタグアドレスを受け取る。そして、テーブルCのマスクデータ28aに新しい部分エントリ26のフラグを追加し、アドレステーブル28bに新しいエントリTタグを書き込む。テーブルD以降のテーブルは常に新規なので、空テーブル管理キュー69に対して新たなテーブルのアドレスを要求し、マスクデータ28aおよびアドレステーブル28bにデータをの書き込む。同時に、バイパス値BPも更新する。をおこなっていく。

【0135】

図39に分類装置15における照合器パイプラインでみた、待ち合わせ方式でエントリ22を登録する構成を示してある。エントリ22の登録は上位から下位へ照合器パイプラインを進みながら、テーブルを辿っていく。バイパス値BPの更新保留となったら、各照合装置50に対応する待機管理ユニット68に必要な情報を記録したパケットを投入し、次段以降に進む。次段へは、直接の上位段となる親テーブルでの待機有無の情報が伝えられる。バイパス値BPが確定した照合装置50においては、親テーブルで待機有りなら、上位フィードバック（FB

）を発行する。この上位FBと並列に、下位へのエントリ登録のパイプライン処理はおこなわれていく。上位FBは待機管理ユニット68で待ち合わせたパケットと合流し、保留していたバイパス値BPを更新する。保留解除によるバイパス値BPを更新する処理とは並列に、必要であれば待機管理ユニット68から更に上位保留へのフィードバックを行う。

【0136】

図40に、待ち合わせ方式でエントリを登録する処理において、更新処理ユニット60が動作したときの照合装置50の制御的な構成を示してある。DTAGは結果タグ（登録データTAG）を示し、TTAGはテーブルタグを示し、PWFは直接の上位段（親テーブル）での待機有無を示すフラグを示す。コマンド「ADD」は上位から下位への既存のエントリのタグを辿って伝播される。コマンド「ADDNEW」は途中段以降の新規テーブルを確保する。コマンド「ADDUPDT」は上位へのフィードでBPの更新を再開する。コマンド「ADDWCLR」は上位で待機している照合装置50に対するバイパス値BPが不変の通知であり、待機解除に繋がる。

【0137】

照合継続情報33と同様の経路で更新用のデータパケットが照合装置50に伝達される。そして、ロードユニット53により照合タグ34で指示されたアドレスのテーブル28がテーブルバッファ59にロードされる（ステップ120）。コマンドデコーダ51はコマンド37をチェックし（ステップ121）、コマンド37は追加「ADD」から開始される。まず、ステップ122で、マスクを読み出し、ステップ123で、キー値とマスク値により次段が新規かどうかを判断する。次段が新規であれば、ステップ124で、空きテーブルキュー69に対してテーブル28nを要求し、応答を待つ間に、ステップ125で、マスクデータ28aの更新処理をおこなう。マスクデータ28aの読み出し値と、追加する部分エントリ26の値の論理和（OR）を演算し、追加エントリに該当するビット（フラグ）を1にする。

【0138】

テーブルキュー69から空テーブルを確保した応答が得られると、ステップ1

26でその空テーブル28nのアドレスをアドレステーブル28bに書き込み、追加処理を伝播するための出力TAGに空テーブル28nのアドレスをセットする。さらに、ステップ127で、出力するコマンドCMD37を「ADDNEW」にセットする。

【0139】

コマンド37が「ADD」であり、次段が既存であった場合、ステップ128で自テーブル28のアドレステーブル28bから次段のテーブルアドレスを読み出し、出力TAGにセットする。ステップ129で出力するコマンドCMD37は入力コマンドと同じく「ADD」のままである。

【0140】

入力コマンド37が「ADDNEW」であった場合、その段のテーブル28nは前段処理により確保された新規テーブルなので、ステップ130でテーブル初期化をおこなう。これはマスクデータ28aだけの処理でよく、マスク値として該当するエントリビットを1に、他を0にする。初期化以降は、コマンド37が「ADD」で新規エントリであった場合と共通の処理である。エントリの登録を照合装置パイプラインに伝播する過程で、一度、コマンド37が「ADDNEW」、すなわち新規エントリになれば、それ以降の照合装置50に供給されるコマンド37は常に「ADDNEW」であり、上記の処理が繰り返される。また、上位のテーブル28にフィードバックされるバイパス値BPを更新する処理については、コマンド37が「ADDNEW」の場合は常におこなう。

【0141】

コマンド37が「ADD」の場合、すなわち、現段階のテーブル28は既存のテーブルである場合は、ステップ131で、エントリの位置によって判断する。コマンド37が「ADD」で、該当（新規）のエントリ26が新規（次段TBLが新規）のエントリ22の部分エントリで、それが最大エントリであった場合（MAXエントリ更新）、バイパス値BPは更新が確定する。コマンド37が「ADD」で該当（新規）のエントリ26が新規のエントリ22の部分エントリで、最大エントリではない場合、バイパス値BPは不変で確定する。コマンド37が「ADD」で該当エントリ26が既存で最大エントリではない場合、バイパス値BPは不変で

確定する。バイパス値BPが確定する場合、更新するのであればステップ132で更新し、ステップ133で出力PFWを0にセットする。一方、コマンド37が「ADD」で該当エントリ26が既存で最大エントリの場合、下位の状況によりバイパス値BPの更新が必要かどうか確定するため、待機管理ユニット68へパケットを登録し、バイパス値BP更新を保留する。更新を保留した場合、ステップ134で、出力PFWを1とする。

【0142】

更新不要、更新確定の場合、状態が定まったので、入力PWFを見、上位待機があるかどうかを判断する。上位待機があれば、1段上の待機管理ユニット68へフィードバック出力する。更新確定なら、ステップ135でコマンド「ADDUPDT」、更新不要なら、ステップ136でコマンド「ADDWCLR」とし、上位の更新要不要を指示する。

【0143】

図41に待機管理ユニット68の動作をさらに詳しく示してある。待機管理ユニット68は、待ち合わせ処理と、その後のバイパス値BP更新再開および上位フィードの並列化をおこなう。コマンドを見て「ADDUPDT」なら下位でバイパス値BPが更新されたので、ステップ137で、バイパス値BPの更新を再開する。そのために照合器50のキューにパケットが投入される。コマンドが「ADDWCLR」なら下位でバイパス値BPの更新がなかったので、発火したパケットは破棄される。ただし、次の上位フィード処理はおこなう。また、コマンドによる処理とはまったく独立に、PWF（上位で待ちがあるかどうか）をみて、PWFが1なら上位へフィードする。そのとき、ID、CMDは下位から来たものをそのまま上位へフィードする。

【0144】

照合装置50では、コマンド37が「ADDUPDT」なら待機していたバイパス値BP更新処理を実行する。上位へのフィードは待機管理側がおこなうので、バイパス値BPを更新した後は照合器50ではパケットを破棄する。

【0145】

双方向リンク方式による登録

図42に、双方向リンク方式による登録の概要を示してある。この方式でのバイパスへの対応は、上から下に辿りながらエントリ登録をおこない、その際にバイパス値BPの更新が確定したテーブルでは更新処理をおこなう。バイパス値BPの更新が未確定テーブルはそのまま下位へ移動し 待ち合わせ方式による登録、待ち合わせ処理のような待ちは設けない。新規テーブル作成段に到達後、その段以下の新規テーブルの連結と、双方向リンクを辿りながらの上位のテーブルに対してバイパス値BPの確定待ちへのフィードバックを並列におこなう。

【0146】

図42において、テーブルAでは、エントリが既存であり、MAXエントリではないため、下位がどうなっても、このテーブルAでのバイパス値BPの更新はない。テーブルBでは、エントリ既存であり、MAXエントリに該当する。従って、下位での結果次第でBP更新がありえる。下位が確定するまではこのテーブルBのバイパス値BPの更新が必要かどうか確定しないため保留する。テーブルCではエントリ新規であり、その新規エントリがMAXエントリとなる。従ってこのテーブルCでバイパス値BPの更新が必要となる。バイパス値BPの更新が発生したので上位へ通知する（双方向リストを利用し上位テーブルBへ遡る）。遡ってきたテーブルBでは下位からの通知を受けてバイパス値BPを更新し、更に上位へ遡っていく。バイパス値BPの更新の必要がなくなった段で遡りは打ち切りである。一方、テーブルCから下への流れは、待ち合わせ方式による登録と同様に処理待機上位への遡りとは並列に進んでいく。

【0147】

図43に分類装置15における照合器パイプラインでみた、双方リンク方式でエントリ22を登録する構成を示してある。双方リンクを辿って上位にフィードバックするので待機管理ユニットは設けられていない。上位フィードバック（FB）はこの図では専用経路であるが、バス効率に影響がなければ検索バイパス用バス39を共用してもよい。双方向リンクによる上位遡りは検索時のバイパス処理（任意の上位へ接続）とは異なり、常に直接繋がる上位テーブル（パイプライン前段の小照合器）へのフィードバックなので、ここでは専用経路による接続とした。エントリ22の登録は上位から下位へ照合器パイプラインを進みながら、

テーブルを辿っていく。バイパス値BPの更新が確定した照合器50から双方リンクにより上位フィードバック(FB)を発行する。この際、待ち合わせ方式と同様に、上位FBとは並列に、下位テーブルへの辿り(新規テーブルの接続)はおこなっていく。

【0148】

この双方向リンク方式においても、並列処理は下位への流れとバイパス値BP確定後の上位へのフィードバックの2並列である。また、上位フィードバックは完全な並列ではないがある程度パイプライン並列化できる。フィードバックによるバイパス値BPを更新する時に、すぐに上位テーブルリンクを読み出して、バイパス値BPを更新する処理とは分岐して更に上位へのフィードバックを発行することができる。ただし、検索・検索バイパスとの関係で待ち行列数が増えるため、全体のパフォーマンスを検討した上で、上位フィードバックのパイプライン並列化の実装は検討すべきである。

【0149】

図44に、双方向リンク方式でエントリを登録する処理において、更新処理ユニット60が動作したときの照合装置50の制御的な構成を示してある。ステップ120から132の処理は、待ち合わせ方式と共通するので説明を省略する。バイパス値BPの更新をおこなうのは、コマンド37が「ADDNEW」の場合と、コマンド37が「ADD」で新規エントリがMAXエントリの場合である。前者の場合は上位へのフィードバックは不要である。コマンド37が「ADD」から「ADDNEW」と切り替わる段のテーブルでフィードバック済のためである。したがって、ステップ140でコマンドをチェックし、後者の場合は、ステップ141で上位リンクを辿り、ステップ142で上位テーブルへ「ADDUPDTコマンドパケット」によるフィードバックを出力する。また、「ADDUPDTパケット」が得られると、ステップ143でマスクデータを読み込み、該当エントリがMAXエントリかどうかでバイパス値BPの更新を判断する。上位から下位への処理でエントリは登録済みであり「ADDUPDT」を受け取った時点では常に既存エントリである。バイパス値BPの更新をおこなった場合、更に上位テーブルへのフィードバックをおこなう。最上位テーブルでは上位リンクがNULLであり、その場合はフィードバ

ックは終了する。

【0150】

待ち合わせ方式による削除

削除も更新処理であり、「排他を確認し、テーブルを占有する」状態でシリアル化されたツリーを構成するテーブルを、テーブル単位で逐次実行される処理である。検索との関係も上記と同様である。削除の場合、登録同様に下位から上位へ伝播するバイパス値BPの更新があり、この上位への遡りを登録同様に待ち合わせ管理機構でおこなうのが待ち合わせ方式による削除である。登録とは異なり削除の場合は上位から下位へのエントリ検索時点ではバイパス値BPの更新を確定できるケースはない。下位テーブルでエントリが0個になった場合にそのテーブルを開放し、上位へ通知し、上位テーブルのエントリを削除し、必要であればバイパス値BPを更新する処理の流れとなる。この処理は下位から上位へ伝播していく処理である。登録と比べるとバイパス値BPの更新は同じであるが、テーブル削除の上位伝播処理があるため、削除のほうが複雑になる。

【0151】

図45に示すように、待ち合わせ方式では、コマンド37が削除「DEL」のときで上位から下位へ辿りながら各段で待機し、最下位から上位に向かってコマンド37を「DELUPDT」としてバイパス値BPを更新し、テーブルを削除した情報をフィードしていく。この基本的な流れは待ち合わせ方式の登録とほぼ同様である。削除の場合、下位でテーブルが空になり開放されると、上位でのエントリ削除とそれに伴うバイパス値BPの更新が発生する。たとえば、テーブルが空になると最終の照合装置50がパイプラインから切り離される。テーブルが空ではない場合は、下位でのバイパス値BPの更新情報を上位へ反映するだけでよい。テーブルが空になった場合、下位でのテーブル削除が上位でのエントリ削除につながり、上位でのエントリ削除の結果、バイパス値BPの更新のため新しいバイパス値BPを与えるエントリの下位への問い合わせするバイパス値問い合わせが発生する。自テーブル内の情報では新しいバイパス値BPを与えるエントリのバイパス値BPは得られないからである。このコマンドが「DELUPDT」のときの下位へのバイパス値問い合わせ（BP問い合わせ）は、登録ではなく削除に特有の処

理である。

【0152】

下位へのBP問い合わせは以下の流れで行われる。ステップ151でエントリ削除にともなうバイパス値BP更新の発生した段で下位へコマンド「DEL RDBP」で問い合わせ、応答待ちの packets を待機する。下位の照合器50では、コマンド「DEL RDBP」を受けると自テーブルのバイパス値BPを読み出し、ステップ152で上位へコマンド「DEL WRBP」を送ってフィードバックする。応答待ちとの間で発火し、コマンド「DEL WRBP」の処理として問い合わせで得られた新しいバイパス値BPを書き込む。さらにステップ153で、バイパス値BPの更新情報を上位へコマンド「DEL UPDT」でフィードバックする。

【0153】

図46に、待ち合わせ方式でエントリを削除（コマンドが「DEL」）する処理において、更新処理ユニット60が動作したときの照合装置50の制御的な構成を示してある。まず、ステップ160で、入力パラメータTTAGで示されるテーブルをロードし、マスクデータ28aを読み込む。ステップ161で、マスクデータ28aと削除対象のエントリ（部分エントリ）をチェックし、対象のエントリが未登録ならステップ162でコマンド37を「DELERR」として削除結果を最終出力する。

【0154】

ステップ163で次段のタグ情報を読み取り、ステップ164で自テーブルが最下段であるか否かを判断する。自テーブルが最下段ではなかった場合は、ステップ165で、下位からのフィードバック待ち合わせ用の packets (ID, WID=1, TTAG (自テーブルTAG), KEY (自テーブルエントリKEY) を含む) を待機管理ユニット68に投入する。ステップ166で、次段の照合器50へコマンド「DEL」(ID, TTAG=Next TAG, KEY=SHIFTを含む) を出力する。

【0155】

一方、自テーブルが最下段であった場合は、ステップ167で、エントリに対応するNextタグ、すなわち、結果タグを読み出し、削除結果として最終出力

する。バイパス値BPの更新も含めた削除完了前に応答が戻るのがまずいようであれば、結果タグは上位フィードバックに載せてフローさせ、最上位まで戻りきったところで最終出力する。次に、ステップ168でエントリを削除する。本発明においては、テーブル28のマスクデータ28aのエントリを削除するだけでよい。エントリを削除した後のマスクデータ28aの積算（和集合）SUM-MASKが0ならばフラグCEFを1（テーブルが空エントリ）にセットし、SUM-MASKが1ならばフラグCEFを0（テーブルにまだエントリが残っている）にセットする。また、エントリ削除前のマスクデータ28aと、削除後のマスクデータ28aを比較し、フラグNBPを設定する。削除したエントリがバイパス値BPを与えるエントリであれば、フラグNBPを1すなわちBP更新有りにセットする。

【0156】

ステップ169でフラグCEFが1である判断すると、自テーブルが空エントリになるので、ステップ170で自テーブルを空きテーブル管理キュー69に返却し、パケットのフラグNBPとDTAGをDC（ドントケア）にセットする。

【0157】

一方、フラグCEFが0の場合、ステップ171でフラグNBPをチェックし、ステップ172で、残ったエントリの中から新しいバイパス値BPを与えるエントリ（MAXエントリ）を選択し、ステップ173で、バイパス値BPを新しいBPエントリ対応するタグ情報（最終結果情報（結果タグ））に更新する。そして、パケットのフラグNBP（バイパス有効）を1にセットし、DTAGにバイパス値BP（新バイパスTAG値）をセットする。なお、ステップ171で自エントリがBPエントリでなかった場合、バイパス値BPの更新処理は不要である。そして、ステップ174で、コマンド「DELUPDT」により上位フィードバックの発火用パケットを出力する。

【0158】

図47に、コマンド37が「DELUPDT」のときの照合装置50における更新ユニット60の処理を示してある。この処理は、コマンド「DEL」が上位から下位に流れたときに待ちに入った照合装置50が、最下段からのコマンド「DELUPDT

」を受けたときに、それをトリガとして発火する。まず、ステップ180でテーブルをロードする。ステップ181で、下位からフィードバックされたフラグCEFが0ならば下位テーブルは消えていないので、自テーブルのエントリ削除は不要である。フラグCEFが0のときは、ステップ182で、下位からのフラグNBPをチェックし、下位テーブルのバイパス値BPが変更になっているときは、さらにステップ183で自テーブルのマスクデータ28aと削除対象のエントリに従い、自己のフラグNBPをセットする。そして、自NBPが1、すなわち、自テーブルのエントリがBPエントリ(MAXエントリ)であれば、ステップ186でバイパス値BPを更新する。下位のフラグNBPが0であれば、ステップ184で自NBPも0にセットしてバイパス値BPの更新は行わない。そして、ステップ187で自段が最上位であるかチェックし、最上位でなければ、ステップ188で上位の待ちへフィードバックする。

【0159】

一方、下位からのフィードバックされたフラグCEFが1ならば下位テーブルが消えたので、ステップ189で、自テーブルのエントリ削除をおこなう。フラグCEFが1のとき、下位テーブルが消えたのでNBPおよびDTAGはDCである。自テーブルのエントリ削除にともなうバイパス値BPを更新する。さらに、エントリ削除後に自テーブルが空(SUM-MASK=0)ならば、ステップ190でフラグCEFOutを1にセットし、ステップ191で自テーブルを開放し、ステップ192で最上位であるか確認後、ステップ193で上位の待ちへフィードバックする。ただし、自テーブルが最上位テーブル(ルートテーブル)である場合、テーブル削除はおこなわない。ルートテーブルは空でも削除しない。また、ルートに対して上位の待ちは存在しないためフィードバックは行われない。

【0160】

エントリ削除後に自テーブルが空ではなかった場合、すなわちフラグCEFOutが0のときは、ステップ194で、新BPエントリ(MAXエントリ)を選択し、ステップ195および196で新BPエントリに対応する下位テーブルからのバイパス値BPの読み出しをおこなう。これは照合器パイプラインの都合に

より、次小照合器50によるバイパス値BP読み出しを待ち合わせ管理機構で待つ方式でおこなわれる。待機側は、待ち合わせパケット（ID，WID=2，TAG，KEY）を投入し、下位への依頼側は、コマンド「DELRDBP」により発行する。MAXエントリでない場合は、ステップ197でルートであるか否かを確認し、ステップ198で上位へフィードバックする。

【0161】

図48にコマンド「DELRDBP」と「DELWRBP」のときの処理を示してある。上位に対するコマンド「DELUPDT」からの指示で下位にコマンド「DELRBP」が送られてくる。コマンドが「DELRDBP」であれば、ステップ201でバイパス値BPを読み出し、ステップ202でDTAGにセットする。そして、上位はバイパス値BP読み出しを待っているの、上位の待機管理68に対して待ち合わせ発火パケット（ID，CMD=DELWRBP，DTAG=BP，WID=2）を投入する。

【0162】

自器のコマンド「DELUPDT」からの待機パケットと下位からのコマンド「DELRDBP」完了後のパケットにより発火するコマンド指示が「DELWRBP」である。下位からのDTAGにバイパス値BPが入っているので、ステップ203では、自テーブルを表すTTAGに従いDTAG値を書き込む。自テーブルにおいてバイパス値BPが確定したので、ステップ205で、コマンド「DELUPDT」による上位フィードバックを再開する。ただし、コマンド「DELUPDT」と同様に、ステップ204でルートテーブルであるか否かをチェックし、ルートの場合は上位がないため、上位フィードバックはおこなわない。

【0163】

双方向リンク方式による削除

この方式では、まず、コマンド「DEL」により上位から下位にエントリを完全一致型の検索で辿り、最下位まで降りていく点は変わらない。コマンド「DELUPDT」による最下位から上位へのバイパス値BPの更新、テーブル削除情報のフィードは双方向リストによるフィードでおこない、待ち合わせ方式のような待機機構は用いない。したがって、図45に示した待ち合わせ方式とほぼ同様のフロー

になるが、上位に対するコマンドは待機管理ユニットではなく照合装置 50 に供給される。

【0164】

また、下位でのテーブル削除にともなう上位でのBP更新については、待ち合わせ方式の場合同様にコマンド「DELRDBP」および「DELWRBP」でおこなうが、この処理も待ち合わせではなく双方向リストによるフロー移動でおこなう。したがって、待ち合わせ方式であれば、自テーブルのKEY値（削除対象のエントリ）は待機パケットが保持していたため下位からのフィードバックはKEYを気にしなかったが、双方向リスト方式ではコマンド「DELUPDT」による上位フィードごとに該当段のKEYを再び使用できなければならない。従って、上位から下位へのコマンド「DEL」による移動でKEYを消費してしまう（消してしまう）のではなく、KEY値はSHIFTによる参照位置移動とし、コマンド「DELPDT」のときには逆方向SHIFTで該当段KEY値を得られるようにしておく。つまり、KEY値は4ビット8段テーブルであれば、32（4×8）ビットのKEYフィールドを持ち、コマンド「DEL」では上位ビットから下位ビットへの参照位置移動、コマンド「DELUPDT」では下位ビットから上位ビットへの参照位置移動をおこなう。

【0165】

したがって、コマンド「DELUPDT」のときは、双方向リストの上位リンクにより上位テーブルタグは得られる。前述のようにKEYは参照位置の移動により、常に該当段のキー値をアクセスできる構造である。コマンド「DELUPDT」のときの下位へのバイパス値BPの問い合わせの際も同様に、上位リンクによる上位テーブルタグの取得、KEY値の適切な参照位置移動がおこなえる。したがって、下位へのバイパス値の問い合わせは、双方向リンクを用いて待ち合わせ方式とは同様に行われる。

【0166】

図49に、双方向リンク方式でエントリを削除（コマンドが「DEL」）する処理において、更新処理ユニット60が動作したときの照合装置50の制御的な構成を示してある。待機管理ユニットを用いない方式なので、図46に示した制

御系統図からステップ165が除かれた以外はほぼ同じ制御で処理は行われる。ステップ174で、コマンド「DELUPDT」で上位フィードパケットを出力する際、送出先のタグアドレスTTAGは上位リンクにより上位テーブルを指定し、エントリKEYは上位方向へ逆シフトし、テーブル段位置と整合させる。

【0167】

図50に、コマンド37が「DELUPDT」のときの照合装置50における更新ユニット60の双方向リンク方式の処理を示してある。この処理においても、待機管理ユニットを用いない点を除けば、図47と同じ制御で処理は行われる。なお、フィードバックする際に、送出先のタグ情報TTAGは自テーブルの上位リンクにより上位テーブルを指定、エントリKEYは上位方向へ逆シフトしてテーブル段位置と整合させることは上記と同様である。

【0168】

図51にコマンド「DELRDBP」と「DELWRBP」のときの双方向リンクによる処理を示してある。これらの処理においても、待機管理ユニットを用いない点を除けば、図48と同じ制御で処理は行われる。

【0169】

上記のように、本発明においては、エントリの追加および削除に伴うテーブルの管理にソーティングや正規化などの処理が不要となるので、エントリの追加削除に伴う照合処理の遅延は従来に比べて非常に小さなものになる。範囲検索におけるエントリとルールとの関係については、右倒れ近似および左倒れ近似の説明において詳述している。たとえば、既存エントリが $[a_i, \text{無限大}] [b_{i+1}, \text{無限大}]$ $\{i \text{ は複数}\}$ の時、新エントリ $[a', \text{無限大}] [b'+1, \text{無限大}]$ を加えるとき（登録する前に）、 $[a', \text{無限大}]$ のルールには a' でヒットするエントリの（複数の）ルールを加え、 $[b'+1, \text{無限大}]$ のルールには $b'+1$ でヒットするエントリの（複数の）ルールを加える。既存エントリが $[a_i, \text{無限大}] [b_{i+1}, \text{無限大}]$ $\{i \text{ は複数}\}$ の時、 $[a_j, \text{無限大}] [b_j+1, \text{無限大}]$ を削除するとき（削除した後でに） $[a_j, b_j]$ に含まれるエントリから該当ルールを除く。

【0170】

データキャッシュ付き分類装置

図52に、データキャッシュ17を備えた分類装置15の概略構成を示してある。データキャッシュ17は、照合器パイプライン500を構成する各照合装置50とメモリであるSDRAM11の間に設置される。図53に示すように、本例のデータキャッシュ17はNウェイ・セット・アソシアティブ・キャッシュであり（Nウェイは2ウェイや4ウェイが一般的である）、キャッシュラインサイズがSDRAM11のバースト転送サイズ、すなわち、照合テーブル28のサイズと一致するように構成されている。データキャッシュ17のキャッシュライン17aを1テーブルサイズとすることで、分類装置15におけるテーブルアクセスが効率的に行われる。

【0171】

また、データキャッシュ内にライン単位のメモリ排他管理機構を導入することで、キャッシュメモリ上あるいはSDRAM上のテーブルデータの排他利用が可能となる。そのためには、キャッシュのラインサイズ、ライン数、セット数を適切に構成し、その構成に合わせてテーブルを割り当てれば良く、データキャッシュヒット率を高めることができる。そして、キャッシュシステム17を採用することにより、汎用のSDRAM11を用いながらさらに高速な分類が可能となる。

【0172】

図54に、データキャッシュ17を検索種別および照合段階で排他的に利用する例を示してある。すなわち、本例のキャッシュシステムにおけるテーブル28のオンキャッシュ率を高めるために、照合テーブル28の各段（たとえば、テーブルA-1とA-2）を異なるキャッシュエントリに割り当て、また異種の照合（たとえば、テーブルA群とテーブルB群）に対してもそれぞれ独立したキャッシュエントリを割り当てる。検索種別は、上流の制御ユニットあるいはアプリケーションから与えられるルートアドレス31により簡単に識別できる。また、照合段階は、パイプライン段数に相当するので、識別は容易である。そして、照合テーブルの新規登録においても、テーブルの確保が必要になった場合、テーブル段位置に対応するキャッシュエントリにマッピングされるSDRAM11のアド

レスを持つ空きテーブルを割り当てていく。

【0173】

また、本発明で用いられる照合テーブル28はツリー型テーブルの構造となるので、下位段でテーブル数が広がっていく。したがって、下位段テーブルには複数キャッシュエントリを割り当てておくことが望ましい。図54においては、照合A用のテーブルA群と照合B用のテーブルB群がSDRAM11の上にある。照合Aの第1段テーブルA-1はキャッシュエントリ0にキャッシングされるアドレスに確保される。照合Bの第1段テーブルB-1も同様にエントリ0になるアドレスである。これらはキャッシュエントリが複数セットで多重化されているため、同時にオンキャッシュとなりうる。

【0174】

同様に照合Aの第2段テーブルA-1はキャッシュエントリ1、照合BのB-2にはエントリ2が対応する。この場合、複数の照合がキャッシュエントリで多重化されず、それぞれ独立したキャッシュエントリが割り当てられている。キャッシュが4セットであれば、A-2であれば全16テーブルでキャッシュエントリ1の4セットを所有し、最新4テーブルがオンキャッシュである。下位段、特に、最終段のテーブルA-8およびB-8は数が多くなるので、1段に対して複数エントリを割り当てることも有効である。

【0175】

図55に、空テーブル管理キュー69が、メモリ11の上でテーブル28を管理する一例を示してある。検索種別毎にあるいはテーブル段毎にキャッシュエントリを分散配置させることでオンキャッシュ率を高めるためには、テーブル28を管理する側、すなわち管理キュー69が分類装置50のテーブル確保要求に応じて適切なテーブルを割り当てる必要がある。空きテーブルキュー69は空きテーブルを検索種別（レイヤー1）とそれに含まれるテーブル段位置（レイヤー2）の2階層に分類した個別キューで管理する。分類装置15の初期化時に、キャッシュエントリ毎にテーブル28をグループ化し、更にROM等に設定された（あるいは外部からコンフィグレーションとして与えられる）（検索種別数×テーブル段数分の）空きテーブル個別キューとキャッシュエントリの対応に従い、テ

ーブルをキューにつないでいく。

【0176】

図56に空きテーブルキュー管理機構69の応答を示してある。空きテーブルキュー管理機構69は、検索テーブルツリーへのエントリ登録時に照合装置50からくるテーブル確保要求(ステップ211)に応じて、空きテーブルを取り出し応答する。この際、検索種別とテーブル段位置が必要となるため、要求情報としてそれらを要求パケットに持たせることとする。管理機構69は、要求情報の検索種別(レイヤー1)・テーブル段位置(レイヤー2)に対応するキューから空きテーブルを1個取り出し(ステップ212)、応答としてそのアドレス(タグ情報)を照合装置50に返す(ステップ213)。

【0177】

図15に示したメモリ共有型の分類装置15においてSDRAM11と照合装置50の間にデータキャッシュ17がある場合、データキャッシュ17の一つがミスヒットするとデータキャッシュ17およびSDRAM11はこのアクセスに占有されて他のアクセスは待たされる。図16に示したメモリ分離型の分類装置15においてもSDRAM11と照合装置50の間にデータキャッシュ17がある場合も同様に、照合装置50、データキャッシュ17およびSDRAM11はこのアクセスに占有される。これらはデータキャッシュ17とSDRAM11が同期的(シリーズ)に接続しているためである。そのため、1つ遅れると、その遅れは後段に影響する。パイプラインの効率を上げるには全てのステージにかかる時間を同一にしてやらねばならないが、SDRAMアクセスが入ると難しい。

【0178】

これに対し、要求応答方式のデータキャッシュ17を採用することができる。すなわち、被検索キー21に対して入力順に照合結果が出力されるというパケット順序の維持はバックトラックが発生することを前提とした段階で不可能であり、識別情報(ID)32を導入することにより順序の問題はクリアしている。したがって、ある被検索キー21の照合ためにテーブル28が要求されたときに、オンキャッシュでなければ照合装置50にその旨を通知する。照合装置50は、処理を中断し、待ち行列に戻し、次の被検索キー21のパケット33を入れる。

データキャッシュ 17 は SDRAM 11 へアクセスする。一方、照合装置 50 からテーブル 28 を要求されたときに、オンキャッシュならば照合装置 50 にテーブル 28 を渡し、照合装置 50 が処理を進める。さらに、照合装置 50 において、照合タグ情報 34 を先読みしてキャッシュシステム 17 に渡し、必要なテーブル 28 をプリロードするシステムは有効である。さらに、照合装置 50 が照合のためにテーブル 28 をバッファ 59 にロードした時点で、次の照合タグ 34 を入れてオンキャッシュを確認することも有効である。

【0179】

このように、本例のキャッシュシステム 17 においては、検索種別および照合段階という要素でキャッシュラインの排他性を高めてオンキャッシュ率を高めている。従来の CPU でのデータキャッシュにおいては、まず、キャッシュラインサイズをテーブルサイズに一致させるという発想はリニアアドレス空間を否定するので従来のキャッシュシステムからは得られないものである。さらに、照合テーブル各段を異なるキャッシュエントリに割り当てるという発想も、論理アドレス（のインデクス部分）とテーブルの割付決定を実行時に行うことであり、OS の元では CPU で実行することは極めて難しい。CPU では論理アドレスとテーブルの範囲割付決定をコンパイル時に行い、実行時には決められた範囲内から 1 テーブル分を確保するだけである。この実行時メモリ割付支援機構が認識するのは先頭論理アドレスとサイズである。動的メモリ割付は確保、開放を繰り返すとメモリに隙間を生じ、これをガベージコレクタで整理するというのが前提であった。従い、インデクス部分の衝突を避けて、キャッシュ効率を上げる仕組みを入れ込めなかった。

【0180】

これに対して、本発明における照合方法および照合装置においては、テーブルビット長を固定化（上記の例えでは 4 ビット）することによってテーブルを標準化したため、テーブルサイズとキャッシュラインサイズとを一致させることが可能になった。そして、キャッシュメモリ 17 や SDRAM 11 に対して検索種別および照合段階で排他的なエリアを割り付けたとしても、テーブル同士は照合タグ情報によりリストでつながれているので、エントリの（実行時の）削除、登録

に対応することができ、オンキャッシュ率を高めるために排他的な領域設定をすることが他の処理において障害となっていないといえる。

【0181】

履歴キャッシュ付き分類装置

図57に、履歴キャッシュ18を備えた分類装置15の概略構成を示してある。照合装置50の前に履歴キャッシュ18をおくことで、検索処理をさらに高速化でき、検索処理の遅延時間を短縮できる。説明してきたように、本発明の照合方法および照合装置、さらに、分類装置および検索装置は、簡易なハードウェアでビット長の増加にも対応でき、照合速度も早く、さらに、汎用メモリを利用して十分な処理速度が得られるといった多くのメリットがある。しかしながら、複数の照合装置をパイプライン状に接続するか、あるいは照合装置を繰り返して使用することにより照合結果を導くので、同一の被検索ビットストリングが高頻度で表れるようなジョブにおいては、CAMあるいはTCAMと比べたときにレイテンシの面で優位とは言えない。したがって、レイテンシが重要となる検索においては、履歴キャッシュ18を用いるメリットがある。

【0182】

図57に示した履歴キャッシュシステム18は、SDRAM共用型履歴キャッシュシステムであり、照合テーブル28の置かれるSDRAM11を共用する構成である。図58に示すように、キャッシングアルゴリズムとして8ビット化ハッシュ+LRUを使用することができる。そのため、本例のキャッシュシステム18は、ハッシュテーブル検索器18aと、ハッシュ用のキー括比較器18bとを備えている。ハッシュテーブル18cは照合装置50と同様に4ビット（16エントリ）×2段のツリー構造で構成し、照合装置50と履歴キャッシュ18の間でデータキャッシュ利用方法の整合性を取るようになっている。

【0183】

履歴SRAMキャッシュ付き分類装置

図59に、履歴キャッシュ専用のSRAM13を備えた分類装置15の概略構成を示してある。履歴キャッシュ18のメモリを専用化した構成であり、テーブル28を記憶したSDRAM11と分離することにより、照合装置50との競合

を防止して照合速度の向上を図ると共に、ZBL-RAMやZBT-RAM等の応答性のよい低遅延SRAMを履歴キャッシュ用に占有して履歴キャッシュ18の性能を向上する。キャッシングアルゴリズムは8ビット化ハッシュ+LRUを使用できるが、共用型の場合は照合テーブルとの整合性のあるメモリブロック構成が望ましかったのに対し、図60に示すように、専用SRAM型の場合は、メモリビット幅やエントリの構成を履歴キャッシュに合わせて最適化することができる。

【0184】

論理演算装置（AND化器）の構成

上述した検索装置10あるいは分類装置15は論理演算装置（AND化器）19を備えている。本発明に係る分類装置15は、汎用型なので、ルートタグ31により検索種別を上流の制御装置あるいはアプリケーションによって設定することができる。したがって、1つまたは複数の分類装置15による検索種別の異なる検索結果（1次元範囲の検索結果）を複数用いて、AND化器19で論理積を演算することにより、多次元範囲の検索を行うことができる。本例のAND化器19においては、1次元範囲の検索結果を集合として取り扱い、行列演算により多次元範囲の検索結果を単純な操作で得られるようにしている。

【0185】

まず、集合Aが要素 $\{A_1, A_2, \dots, A_n\}$ で構成されたとする。要素 A_i がバイナリ（0/1）のmbit列 “ $a_{i1}, a_{i2}, \dots, a_{im}$ ” で表せるとする。同様に集合Bが要素 $\{B_1, B_2, \dots, B_n\}$ で構成されたとする。要素 B_i がバイナリ（0/1）のmbit列 “ $b_{1i}, b_{2i}, \dots, b_{mi}$ ” で表せるとする。

【0186】

このとき、集合A、Bの積（AND）集合A&Bは、行列におけるA、Bの積 $A \times B$ で表せる。詳しい演算式は、図61に示す通りである。図62に、集合Aが2要素、集合Bが3要素、さらに各要素が3ビットで現れる例を具体的に示してある。左辺行列Aは集合Aの要素 A_1, A_2 が各行に並べられている。また、行列Bは集合Bの要素 B_1, B_2, B_3 が各列に並べられている。右辺行列Zの

1行1列は「 $a_{11} * b_{11} + a_{12} * b_{21} + a_{13} * b_{31}$ 」であり、「 $A_1 \times B_1$ 」である。これはAND化器19におけるブール演算の定義により、 A_1 と B_1 のビット比較結果（1＝一致／0＝不一致）を意味する。同様に行列Zを列方向で見ると、1列目は「 A_1 と B_1 の比較結果」と「 A_2 と B_1 の比較結果」になっている。同様に行列Z 2列目は「 A_1 と B_2 」と「 A_2 と B_2 」の比較結果である。つまり、行列Zの各列は行列A全体（集合A）と行列Bを構成する各列（集合B要素 B_i ）の比較結果である。

【0187】

したがって、行列演算の結果、例えば、図63に示すように、右辺行列Zがなっていれば、左辺行列Xの $\{a_{11} \ a_{12} \ a_{13}\}$ 行（集合要素 A_1 ）が集合Aと集合Bの積集合 $A \& B$ である。つまり、行列Zの各行において行内に1つでも1が立つ行に対応する行列Xの行が積集合である。ここで「1つでも1が立つ行」を言い換えれば、行列Zの各行において行内の値の論理和ORを取った結果が1となる行に対応する行列Aの行が積集合である。

【0188】

したがって、行列Aに対して、論理積を求める要素を掛け算し、1つでも1が立つ行が残れば、それらの要素の積集合が得られたことになる。図64にLSI化を念頭に行列演算を一般化したものを示してある。左辺の行列Xは行毎に集合Aの要素 A_1, A_2, \dots である。左辺の行列Yは列毎に集合Bの要素 B_1, B_2, \dots に加え、集合Cの要素 C_i, \dots で構成される。列の数は限定されない。行列Zを列毎にみると、行列X全体と行列Yのある1列との演算となっている。行列Zは最終的には行毎に行内で論理和ORが演算され、 $A * B * C * \dots$ の積集合要素が決定される。従って、行列Zはその値全体を保持しておく必要はなく、列毎にブール行列演算した上で、累積結果としてOR演算していけばよい。

【0189】

図65に、AND化器19のハードウェア構成の一例を示してある。AND化器19は、複数のブール行列演算レジスタ19aを有しており、集合Aの各要素が各レジスタ19aにセットされる。さらに、各々のレジスタ19aに対応して

、レジスタ 19 a の演算結果の累積論理和 (OR) を演算して結果を蓄積するアキュムレータ 19 b を備えている。各アキュムレータ 19 b は 0 に初期化される。この AND 化器 19 に対し、入力要素 19 i n として、集合 B の要素 B i、集合 C の要素 C i などの各列を順次に投入すると、入力列毎に以下の演算処理がおこなわれる。まず、ブール行列演算レジスタ 19 a と入力列による、ブール行列演算 $A_n \times Y_i$ (ENOR 演算、AND 演算となる) が行われる。次に、ブール行列演算の結果とアキュムレータ値による OR 演算が行われる。そして、OR 演算の結果によりアキュムレータ 19 b の値が更新され、最終入力後にアキュムレータ 19 b に 1 が残る行に対応する集合 A の要素群が集合 A, B, C, ... の積 (AND) 集合、すなわち、 $A \& B \& C \& \dots$ である。

【0190】

ブール行列演算は ENOR 論理によるビット比較を AND 論理で繋いだものであり、回路的に見れば、各ブール行列演算レジスタはレジスタ値 A_n と入力値 Y_i との m ビット完全一致比較結果を出力し、その比較結果をアキュムレータで累積 OR していることになる。ブール行列演算レジスタの数やビット長にもよるが、ENOR 論理および AND 論理を組み合わせる回路を作る事は極めて簡単である。そして、入力値 1 個とレジスタ値複数個が一度に比較できるので、レジスタ数分の並列性がある。したがって、本例の AND 化器 19 は、単純な構成でありながら、短時間で多次元範囲検索の結果を導出することができる。あるいは、並列レジスタがコスト高という場合は、レジスタを 1 個あるいは数を減らして、レジスタ値はメモリにし、入力値に対して 1 行分ずつ繰り返しブール行列演算することによっても本例の AND 化器 19 は実現できる。

【0191】

レジスタ値をメモリに展開することにより、メモリの許す限り、ルールを入れることができるというメリットがあり、多数のルールあるいは要素を有する集合から論理積により要素を絞り込むのに適している。ただし、繰り返し処理でありメモリリードもあるため、処理時間は長くなる。ブール行列演算処理とメモリリードをパイプライン化することは可能であり、この構成によりメモリリード処理時間を隠蔽することも可能である。

【0192】

さらに、AND化器19は内部にアキュムレータ19bを持ち、またレジスタ19aに最初の集合の要素を保持する構造のため、複数の多次元範囲検索の並列実行を考えるとそれらは共有資源である。したがって、何らかの資源排他占有が求められる。一方、分類装置15においては、複数検索の各次元の検索が並列におこなわれえるため、例えば、検索系 S_1 、 S_2 の各次元A、Bについて、分類器からの出力は S_1A 、 S_2B 、 S_2A 、 S_1B のように検索系も同検索内の次元も順序性のない出力となりうる。先に分類装置15から出力された検索系がAND化器19を占有利用する方式でも、先に分類装置15からの全次元出力がそろった検索系が占有利用する方式でも、いずれにせよ分類装置15の出力が非同期・無順序であるため、AND化器19の利用率および多次元範囲検索としてのスループットを高効率でおこなうためには、AND化器19も並列化することが望ましい。AND化器19のレジスタおよびアキュムレータをメモリ化した場合は、メモリ上での並列化でも効果が得られる。

【0193】

また、AND化器19の動作を考えたとき、初期設定「BEGIN(レジスタ設定・アキュムレータ初期化)」と終了処理「END(アキュムレータからの読み出し)」が必要であり、同系内でも分類器入力と出力の間での順序性がないため、次元Aだから初期設定処理、次元Bだから終了処理というように操作は固定的に決定できない。分類装置15からの出力を検索系ごとにキューイングし、全次元の分類器出力が完了したのち、明示的に初期設定と終了処理をコントロールする方式となる。結局、これらの制御もあるので、分類装置からの出力がそろった検索系がAND化器19を排他占有する制御方式が望ましいといえる。

【0194】

AND化器19の並列利用のためには、速度を要求されるならAND化器19そのものの複数用意することで対応できる。次善策としては、レジスタ19aおよびアキュムレータ19bのメモリ保存によるメモリ上での並列化をおこなうことで対応できる。さらに、履歴キャッシュ18を導入している場合、多次元範囲検索における履歴キャッシュ18の使い方を工夫することで、AND化器19の

使用率を下げることができ、AND化器19を並列利用しなくても十分な処理速度を維持することも可能である。例えば、32ビット5次元の多次元範囲検索であれば、160ビット(32ビット×5)のキーによる履歴キャッシュ18を採用し、履歴キャッシュ18のエントリデータとして、初回検索時にAND化器19を使って演算した5次元の積集合結果を登録する。つまり、履歴キャッシュ18には分類装置15の最長一致検索の履歴、完全一致検索の履歴、1次元範囲一致検索の履歴をエントリデータとして登録することも可能であるが、多次元範囲検索の場合は、各次元個別の履歴ではなく、全体としての多次元範囲検索の履歴をエントリデータとして取り扱うことによりAND化器19による処理がクリティカルパスになることを抑制できる。

【0195】

上記に、幾つかの例を示しながら本発明についてさらに詳しく説明しているが、上記の例に本発明が限定されないことはもちろんである。たとえば、照合テーブル28の数をさらに削減できる方法がある。照合装置50を用いた検索においてはエントリビット列を4ビットずつの部分エントリに分割してテーブル化し、テーブル化ビットツリーにより登録し検索していく。この方法では、ある段以降が1エントリしか存在しなくてもテーブルを生成し、検索においてもテーブルを辿る必要がある。これを改善し、ある段以降が1エントリであれば下位テーブルを省略するテーブル構造を導入することが可能である。

【0196】

図66に下位テーブルを省略する例を示してある。テーブルBは本来のテーブル構成ではB-C-Dとなる1エントリが存在するが、下位が1エントリであるためC-Dを省略し、その代わりにテーブルBに省略情報(エントリの省略した下位ビット列、省略段数、省略マークなど)を記録しておく。テーブルEでは直接の下位接続はテーブルFの1個であるが、エントリとしては孫テーブルで分岐したGおよびHの2エントリ以上が存在するため、省略はされない。

【0197】

テーブルBの下位に収容される2エントリ目の登録時に、省略された下位テーブルは展開(生成)される。上図ではB-C-Xと繋がるべきエントリXの登録

が行われる際、テーブルBにて下位テーブルの省略情報を検出し、第一エントリに相当するテーブルC-Dの生成、第二エントリに相当するテーブルXの生成をおこなう。

【0198】

ここで、第一エントリのテーブル生成と第二エントリのテーブル生成はシリーズにおこなうのではなく、テーブルBの次の段は同ビット列なので共通テーブルCを生成し、テーブルCの次では異なるビット列なので分岐しテーブルDおよびテーブルXを生成するというように、2エントリが分岐する段までは統合的にエントリを生成することができる。したがって、エントリを登録する処理時間を短縮できる。

【0199】

また、分岐の次段では再度それぞれの下位が1エントリになる。このため、そこで省略テーブル形式でテーブル生成することになる。例えば、上図においてテーブルDおよびXが最下段ではなく、さらに下位に延びているのであれば、テーブルCで分岐した次のテーブルDおよびXはその下位にそれぞれ1エントリしか存在しないため、下位テーブルは省略される。省略テーブル形式で生成されると、検索時は照合装置50がテーブル28の省略マークを見て、そのテーブル下位が省略されていることを検出し、次段のテーブルを読み出さずに最終照合結果を得ることができるので、照合時間も短縮できる。

【0200】

圧縮テーブル（照合テーブル）28には、4ビット×n段分のエントリ省略ビット列が記録されているので、照合装置50では、エントリビットをシフト読み出ししながら検索モードに応じた照合を行う。1エントリに対する完全一致・近似一致の照合であれば、上述した照合器50の比較部52を使う代わりに、専用化した比較器を実装したほうが効率的である。また、専用比較器を使用するのであれば、4ビット比較×n回でおこなう必要はなく、リソースのバランスを見た上で多ビットの比較器とすることが望ましい。

【0201】

省略テーブルでは被検索キー21と1エントリとの比較であるため、完全一致

検索であれば完全一致照合、範囲検索であれば、大・同・小が判定できる比較照合が可能な比較器でよい。また、比較後のエントリ選択処理は不要であり、比較結果から検索結果を判定するだけである。完全一致であれば比較で一致なら検索成功、不一致なら検索失敗。範囲一致であれば比較で大または一致であれば（右倒れ型）検索は成功、左倒れ型の場合も同様に比較が小または一致なら検索成功である。

【0202】

削除については、削除の結果、1 エントリとなる部分を検出し、その下位からエントリビット列を集約し再度の省略テーブル化を行わなければならない。テーブル管理の処理時間は増加する可能性があるが、上述したように、照合時間を短縮できるので、メリットは大きい。各段のテーブル 28 に収容エントリ数カウンタを設け、上位から下位へ削除エントリを辿る際にカウンタを減算し、1 エントリのみとなるテーブル位置を検出する。そのテーブルを省略テーブルに変更し、その下位の省略対象エントリのテーブル列を辿り、最下段から上位へと省略ビット列（＝エントリ No）を合成しながら上位へと上げていき、省略テーブルに集約して省略ビット列として記録するという方法がある。削除によるテーブル再省略化は、上述した更新処理よりも複雑でありテーブルアクセスも増える（時間がかかる）。従って、登録時のみ省略・展開をおこない、削除では再省略化をおこなわない方式もありうる。（検索頻度と登録・削除頻度とのバランスで、登録削除が一定以上多い場合、展開・省略の繰り返し、すなわち、テーブル確保・開放の繰り返しとなり、分類器全体の性能低下を招く可能性もある。

【0203】

図 67 に、要求応答方式のメモリアクセスと小分類器内部のパイプライン化の概要について示している。上記においても説明したが、照合装置 50 においてはマスクデータ 28a の読み出しがボトルネックの 1 つである。マスクデータ 28a がなければ、判定ユニット 54 から下流の処理（VLD から SEL および出力ユニット 55 における処理）が実行できず、マスクデータ 28a の読み出しと並列に実行できるのは主に比較ユニット 52 の処理だけである。従って、照合器 50 の内部処理もパイプライン構成とし、テーブルロード（マスク読み出し）フェ

ーズにてキャッシュ (SDRAM) へメモリリード要求を出し、応答のあったものから VLD 以降の処理をおこなうことが考えられる。そのために、VLD ステージの前に待ち行列 57 があり、そこで応答の待ち合わせがおこなわれる。

【0204】

図 67 に示した例では、メモリ要求～応答の間に CMP ステージ 52 を入れているため、待ち行列 57 に待ち合わせ機構が必要となる。データキャッシュなしの場合は、メモリアクセスは FIFO であり待ち行列 57 も FIFO による待ち合わせでよいが、データキャッシュがあった場合、ヒット・ミスヒットにより順序性がなくなり、CAM 等の FIFO 型よりは高度な待ち合わせ機構が必要となる。データキャッシュなしの場合でも、メモリアクセスをとまなわれないモード (スルーなど) があるので、やはり単純な FIFO 待ち合わせ型の行列では無理がある可能性がある。メモリアクセスを伴わない場合も、ダミーのアクセス要求を出してダミー応答を戻させ、応答数と FIFO 待ち数を整合させれば FIFO 型で対応できる可能性がある。

【0205】

図 68 に異なる照合装置 50 の例を示してある。この例では、メモリアクセスを伴うものとそうではないものを分岐させ、メモリアクセスを伴うものは非順序の応答を VLD 直前の合流待ち行列 57 に戻す。一方、メモリアクセスを伴わないものは、待ち行列 57 の前のステージ 57p で必要な処理があればおこない、合流待ち行列 57 に入る。この場合、合流待ち行列 57 は単なる 2 方向からの合流のみであり待ち合わせは不要となる。この構成では、待ち合わせ機構を不要とするため、メモリアクセスと CMP 処理 52 の (空間的な) 並列化はなされないが、照合装置 50 の内部のパイプラインにより時間的な並列化には有効である。

【0206】

図 69 に、マスクデータを先読み対応可能なテーブル化によるメモリアクセスストールを解消する例について示してある。マスクデータの読み出しに係るボトルネックを解消するためにテーブル 28 の構成を変更する。図 69 (a) に示すテーブル 28 は、そのテーブルのエントリ有無を示す情報としてマスクデータ 28a があり、次テーブルのアドレスを示すアドレステーブルを示す NextTAG×16

エントリで構成されている。図 69 (b) に示す改良型テーブル 1 は次段のアドレス「NextTAG×16 エントリ」28b の次に、次段のマスクデータ「NextMask×16 エントリ」28p を配置したテーブル構成となっている。また、図 69 (c) に示す改良型テーブル 2 は、次段のマスクデータ「NextTAG」28b と次段のマスクデータ「NextMask」28p のペアが 16 エントリ登録されているテーブル構成である。いずれも、次段の照合テーブル 28 のアドレスだけではなく、マスクデータ 28a も備えているので、次段のマスクデータ 28a をプリフェッチすることができる。

【0207】

ただし、改良型においても、テーブル管理や検索以外での利用のため、次段のマスクデータ「NextMASK」と同じ値が、次段のアドレス「NextTAG」の指すテーブル 28 に記録されていることが望ましい。したがって、あるテーブルにはそのテーブル自身のマスクデータと、次段のマスクデータの 2 種のマスクデータが存在することが望ましい。テーブル下位からエントリを辿っていく場合に、上位のテーブルにしかマスクデータがない場合、そのテーブルだけでは隣接エントリへの移動が不可能になってしまうからである。

【0208】

次段のマスクデータをプリフェッチする構成にするのに伴い、照合装置 50 の間をフローする照合継続情報のデータパケット 33 にマスクデータ 28a を追加する。この改良により、照合装置 50 の処理においては、入力パラメータ（照合継続情報）33 としてその照合装置 50 で使用する照合テーブルのマスクデータ 28a が与えられる。このため、マスクデータ 28a を必要とする判定ユニット 54 から下流の処理と、ロードユニット 53 の処理とを並列に実行することが可能となる。

【0209】

マスクデータ 28a をリードすることに起因によるストールを解消するという点では、分類装置 15 あるいは照合装置 50 に内部 SRAM を設け、各テーブルのマスクデータ 28 を照合装置 50 の内部に配置することで並列度を向上することも可能である。

【0210】

以上に説明したように、本発明に係る照合方法および照合装置は、IPヘッダのようなビットストリングを一致検索したり、範囲検索してルーティングするデータ管理装置1の検索機構10、8s、7sおよび6sとして好適なものである。しかしながら、本発明の照合方法および照合装置の適用範囲は、これらに限定されるものではない。たとえば、複数の照合装置50を備えた分類器15を様々な応用する場合、MIN検索・MAX検索があると便利である。バイパスをMAXバイパスとしていた場合に、被検索キー {11...11} で右倒れ近似検索をおこなうことで、登録エントリの中からMAXエントリを検索することはできる。だが、MINエントリは検索できない。従ってこの場合、MIN検索モードで検索することでMINエントリを取り出すことになる。ソートなどのように検索ではなく登録&取り出しで本発明に係る分類器15を利用できる場面がある。このとき、MIN検索およびMAX検索の両方を備えていることはメリットがある。

【0211】

照合器50において、比較ユニット52の処理を省略し、判定ユニット54からなる検索モードが実行できるようにすることも有効である。たとえば、比較ユニット52のCMP i から h は常に「=」(01)が出力されるようにする。あるいはMIN検索・MAX検索モードになったときには、比較ユニット52をスルーして判定ユニット54のVLD 54 a に「01」が入力される回路としても良い。この検索モードにおいては、VLD 54 a により有効エントリのみ「01」となり、SEL部54 b にて、MAX検索モードであれば比較結果が「01」のエントリの中から最もエントリNoの大きなエントリが選択される。また、MIN検索モードであれば最もエントリNoの小さいエントリが選択される。右倒れ・左倒れのG_L・G_R関連部でエントリ選択論理を「>」(11)・「<」(10)から「=」(01)に変更することにより実現できる。テーブル乗り移り処理であるタグ更新部55 b は完全一致検索モードに準じ、出力S_r・S_lで指定されたエントリNoを次テーブルとした照合タグ情報34の更新をおこなう。以上により、各段のテーブルで最小エントリ・最大エントリを辿っていくMIN検索・MAX

検索機能を提供できる。

【0212】

次に、分類装置15をソートへ適用する場合、ソート後のデータ読み出しのための拡張が必要となる。昇順先頭読み出し・降順先頭読み出しはMIN検索・MAX検索で可能である。従って、昇順次読み出し・降順次読み出しのためのコマンドが必要となる。これを実現するためには、テーブルは双方向リンク型テーブルとする。また、昇順先頭読み出し（MIN検索あるいは{0・・・0} 近似検索）・降順先頭読み出し（MAX検索あるいは{1・・・1}）の出力パラメータとして、最下段テーブルのTAG+エントリNoを読み出しハンドラとして出力する。

【0213】

さらに、コマンド「NEXT・PREV」を設け、読み出しハンドラ（最下段テーブルTAG+エントリNo）を入力パラメータとして受け取ると、次エントリの結果TAGと共に更新した読み出しハンドラ（最下段テーブルTAG+エントリNo）を出力する。照合器50の内部では、まず照合TAG（読み出しハンドラの最下段テーブルTAG）34に従い、テーブル28よりマスクデータ28aを読み出す。読み出しハンドラのエントリNoに基づき、次エントリNo（前エントリNo）を算出し、テーブル28から対応するエントリのアドレス「Next TAG」（最下段なので結果TAG）を読み出す。もし、次エントリNo（前エントリNo）が別のテーブルになる場合は、双方向リンクの上位リンクを使い1段上位テーブルに遡り、そこで隣接エントリに移る。1段上位にも自エントリしかなかった場合は更に上位に遡っていく。上位テーブルで隣接エントリに移ったら再び最下段まで降りなければいけないが、これはMIN検索・MAX検索モードによりおこなう。

【0214】

本発明の分類装置15は、リレーショナルデータベース（RDB）にも適用できる。バイナリビットツリーによる検索はデータベースではインデクスに相当する。したがって、複数のフィールドをバイナリビットツリー登録しておけば、これらをキーとしてデータベースを検索できる。検索の種類は最長一致でも、範囲

一致でも、完全一致でも対応できる。問題は複数のフィールドを検索後のAND処理（積集合をとる処理）である。つまり「 $A_0 < \text{身長} < A_1$ 」と「 $B_0 < \text{体重} < B_1$ 」を別々に検索はできても「 $A_0 < \text{身長} < A_1$ 」かつ「 $B_0 < \text{体重} < B_1$ 」の集合をどう高速に得られるかが問題となる。予めフィールドが固定的に決定されているならば、そのフィールド群による複合（多次元）インデックスを生成しておくのが一般的である。対象フィールドが固定的に決定できず、複合インデックス（身長と体重の2次元インデックス）が生成できない場合、従来のCPU+OSの検索システムではおそらく次の方法で論理積を演算する。すなわち、レコード名を R_i とすれば、「 $A_0 < \text{身長} < A_1$ 」の集合 $\{R_1, \dots, R_m\}$ （個数は m 個）と、「 $B_0 < \text{体重} < B_1$ 」の集合 $\{R'_1, \dots, R'_m\}$ （個数は m' 個）を求め、二つの集合のメンバを総当りで比較し両方に属するメンバを残す。これがAND集合になる。この場合、比較計算は $m \times m'$ 回になる。

【0215】

これに対し、本発明の分類装置を用いると、次のような操作が可能である。すなわち、集合 $\{R_1, \dots, R_m\}$ を分類器15に登録し、集合 $\{R'_1, \dots, R'_m\}$ を通した時、前の集合で登録したメンバにヒットすればこれがAND集合のメンバになる。この場合、比較計算は $m + m'$ 回になり、処理速度が向上する。

【0216】

本発明の分類装置は、積集合（AND集合）だけでなく、和集合（OR集合）の演算も、ユニーク化も、また、ソートも簡単にできる。CPU+OSの検索システムでデータベースにテンポラリーな印をつけて良いならば次の方法で論理積（AND）を演算できる。集合 $\{R_1, \dots, R_m\}$ の各メンバを検索の都度印をつける。 $\{R'_1, \dots, R'_m\}$ の各メンバを検索の都度前の印があるものをあげる。この場合、比較計算は $m + m'$ 回になるが、不要な印の処理やディスク本体のデータベースへのアクセスなどの問題が起こり、時間がかかりすぎる。ビットマップ法は高速に論理積（AND化）できるがルール数に制約があるなど汎用化は難しい。

【0217】

これに対し、本発明の分類装置によるAND集合をとる方法はパケットフィルタリング等の複数ルール集合のAND集合化にも使える。本分類器15では、例えば、5ID（5種類のビットストリームに対する検索）に対応する5インスタンスを並列に動かし、ID毎にヒットするルールに印をつけていく。既に4つの印を確認したインスタンスはこれをANDインスタンスへ通知する。ANDインスタンスは通知される都度、今の最優先ルールと比較し、更新する。5インスタンスが全て終了した後、今の最優先ルールに決定する。

【0218】

本発明の分類装置により、ツリー構造型の非リニアメモリアドレッシングの実現も可能となる。アドレスを検索キーとした形で、本発明の分類器をメモリ管理機構に取り込むことで、上位層からみて柔軟なアドレッシングが行えるメモリ空間が実現できる。これは従来のリニアアドレス空間メモリに対し、上位層が明示的に構造化管理をおこなっていた部分をメモリ管理機構が隠蔽カプセル化し、メモリを抽象化オブジェクト化することに他ならない。

【0219】

従来、CPUのコードでおこなっていたメモリのオブジェクト化アプローチとは異なりハードウェアベースのアプローチなので高速効率的にオブジェクト化が実現される。特に、検索のためのメモリ利用の場合、分類器によるアドレッシング（＝照合）が効果的に利用できる。

【0220】

図70に、その一例を示してある。受信時、I/F230を経由してリニアアドレス（論理アドレス）の書き込みデータ（受信データ）を受け付けると、分類器15は受信データをSDRAMメモリ11へ書き込む。分類器15はアドレス変換機能232を持っているので、論理アドレス（リニアアドレス）から実アドレス（リストやツリーなど何らかの構造をもった非リニアアドレス）へ変換してSDRAM11に書き込むが、I/F230を介してデータを送信している方は、リニアアドレスで書き込むことができる。送信時も同様である。I/F230からリニアアドレスで読み出すときに、分類器15を経由することで、アドレス変換をおこない、リニアアドレスから実アドレスへ変換し、非リニアなアドレス

からの連続読み出しに対応できる。

【0221】

CPU235も分類器15によるアドレッシングを利用できる。CPU235はデータフロー動作ではないため、分類器15が提供するアドレッシング配下のメモリ11だけではコード実行等に支障がある。従って、コード蓄積やデータ加工用として独自にメモリ236を持ち、分類器15の配下のメモリ11はパケットメモリとして利用する。ルータ等を考えた場合、CPU235は分類器15によるTCP/IP処理を補助するものとして、ルーティングエントリ管理やフィルタリングルール管理などのハードウェア処理ではカバーできない処理を肩代わりする。つまり、分類器15がメイン機能を担当し、CPUはサブとして動作する。

【0222】

本発明の分類装置15は、状態遷移によるテーブルルックアップ駆動のプロセッサにも適用可能である。図71は単純状態遷移を示しており、key_iを検索し、その結果TAG(ルール)でkey_i+1を読み取れるようにしておけば、key₁-key₂-...-key_j-key₁のような単純ループ型の状態遷移ができる。

【0223】

図72は、イベント連動型単純状態遷移を示しており、イベント_iがトリガとなってkey_i+1に遷移しイベント_i以外でkey_iに戻る。key_iはイベント_i待ち行列でイベント_iを待つ。イベント_iが来たらkey_iにイベント_iを直列につないで検索する。イベント_iは1ビットで、正が{1}、誤が{0}としkey_iを「101...01」とすれば「101...011」でヒットし、「101...010」でミスヒットとなる。ヒットルールにはイベント_i+1待ち行列アドレス、key_i+1が書き込んである。ミスヒットルールにはイベント_i待ち行列アドレス、key_iが書き込んであり、戻ることになる。さらに、本発明の分類装置は、条件分岐を含む状態遷移にも適用可能である。

【0224】

また、知的連想マシン、オートマトンへの応用(言語理論・オートマトン理論

）、大規模なデータベースから有用なルールあるいはパターンを高速に発見するデータマイニングと呼ばれる分野などでも有効であり、応用範囲は非常に広い。

【0225】

【発明の効果】

以上に説明したように、本発明に係る照合方法および装置は、完全一致検索、最長一致検索から多次元範囲一致検索までカバーでき、複数の照合装置を備えた分類装置は、パケットフィルタリング・SPD (Security Policy Database) 検索、ステートフルインスペクションなどのルール検索に用いられることができる。さらに、データキャッシュ、履歴キャッシュを導入することにより、照合速度を向上することが可能であり、適用範囲は広い。検索以外の応用機能としては、集合抽出、集合演算、整列（ソート）・ユニーク化、メモリアドレッシング、状態遷移マシン、データマイニングなどを挙げることができる。

【図面の簡単な説明】

【図1】 本発明に係る管理装置の概要を示す図である。

【図2】 一致検索の照合状態を説明する図である。

【図3】 バイナリビットツリー上の照合状態を示す図である。

【図4】 右倒れ一致型の近似について説明する図である。

【図5】 左倒れ一致型の近似について説明する図である。

【図6】 バイナリビットツリーをテーブル分割した様子を示す図である。

【図7】 照合テーブル（部分テーブル）を示す図である。

【図8】 MIN/MAXサーチを説明する図である。

【図9】 部分テーブルでMIN/MAXサーチを行う様子を示す図である。

。

【図10】 圧縮テーブルでMIN/MAXサーチを行う様子を示す図である。

【図11】 バイパス処理を示す図である。

【図12】 照合過程でバックトラックが発生する様子を示す図である。

【図13】 バックトラックが発生した後に、バイパス処理を実行する様子

を示す図である。

【図 14】 検索装置の概要を示す図である。

【図 15】 メモリ共用型の分類装置を示す図である。

【図 16】 メモリ分離型の分類装置を示す図である。

【図 17】 繰り返し型の分類装置を示す図である。

【図 18】 照合装置の概要を示す図である。

【図 19】 照合装置の処理を示すフローチャートである。

【図 20】 照合装置における処理の進行を説明する図である。

【図 21】 比較ユニットを示す図である。

【図 22】 比較器の論理を示す図である。

【図 23】 比較ユニットの異なる例を示す図である。

【図 24】 マスク器を示す図である。

【図 25】 マスク器の論理を示す図である。

【図 26】 選択回路の一致論理を示す図である。

【図 27】 選択回路の範囲検索（右倒れ）の論理を示す図である。

【図 28】 選択回路の範囲検索（左倒れ）の論理を示す図である。

【図 29】 比較ユニット、マスク器、選択回路の出力（右倒れ）を示す図である。

【図 30】 比較ユニット、マスク器、選択回路の出力（左倒れ）を示す図である。

【図 31】 コマンドの例を示す図である。

【図 32】 バイパス処理用のバスでバックトラックを処理する様子を示す図である。

【図 33】 メモリ共有型の分類装置でバックトラックを処理する様子を示す図である。

【図 34】 分類装置においてスルーを処理する様子を示す図である。

【図 35】 異なる方式でスルーを処理する様子を示す図である。

【図 36】 さらに異なる方式でスルーを処理する様子を示す図である。

【図 37】 さらに異なる方式でスルーを処理する様子を示す図である。

【図 3 8】 待ち合わせ方式でエントリを追加する処理を示す図である。

【図 3 9】 待ち合わせ方式でパイプラインに沿って処理を進める様子を示す図である。

【図 4 0】 待ち合わせ方式でエントリを追加する更新ユニットの機能を示す図である。

【図 4 1】 待機管理ユニットの機能を示す図である。

【図 4 2】 双方向リスト方式でエントリを追加する処理を示す図である。

【図 4 3】 双方向リスト方式でパイプラインに沿って処理を進める様子を示す図である。

【図 4 4】 双方向リスト方式でエントリを追加する更新ユニットの機能を示す図である。

【図 4 5】 待ち合わせ方式でエントリを削除する処理を示す図である。

【図 4 6】 待ち合わせ方式でエントリを削除する更新ユニットの機能を示す図である（コマンドがDELのとき）。

【図 4 7】 待ち合わせ方式でエントリを削除する更新ユニットの機能を示す図である（コマンドがDELUPDTのとき）。

【図 4 8】 待ち合わせ方式でコマンドがDELRDBPおよびDELRB Pの処理を示す図である。

【図 4 9】 双方向リスト方式でエントリを削除する更新ユニットの機能を示す図である（コマンドがDELのとき）。

【図 5 0】 双方向リスト方式でエントリを削除する更新ユニットの機能を示す図である（コマンドがDELUPDTのとき）。

【図 5 1】 双方向リスト方式でコマンドがDELRDBPおよびDELRB Pの処理を示す図である。

【図 5 2】 データキャッシュを備えた分類装置を示す図である。

【図 5 3】 データキャッシュのラインサイズを示す図である。

【図 5 4】 データキャッシュの管理を示す図である。

【図 5 5】 メモリの管理を示す図である。

【図 5 6】 空テーブル管理キューの応答を示す図である。

【図 57】 SDRAM 共用型履歴キャッシュシステムを備えた分類装置を示す図である。

【図 58】 SDRAM 共用型履歴キャッシュシステムの概要を示す図である。

【図 59】 履歴 SDRAM キャッシュ付き分類装置を示す図である。

【図 60】 履歴 SDRAM キャッシュの概要を示す図である。

【図 61】 AND 化器における行列演算を説明する図である。

【図 62】 行列演算の一例を示す図である。

【図 63】 行列演算の左行列を示す図である。

【図 64】 LSI 化に適して行列演算の概要を示す図である。

【図 65】 AND 化器のハードウェアを示す図である。

【図 66】 下位テーブルを省略する様子を示した図である。

【図 67】 要求応答方式のメモリアクセスを採用した照合装置の概要を示す図である。

【図 68】 要求応答方式のメモリアクセスを採用した照合装置の異なる例を示す図である。

【図 69】 照合テーブルの改良型を示す図である。

【図 70】 非リニアメモリアドレッシングを示す図である。

【図 71】 単純状態遷移を示す図である。

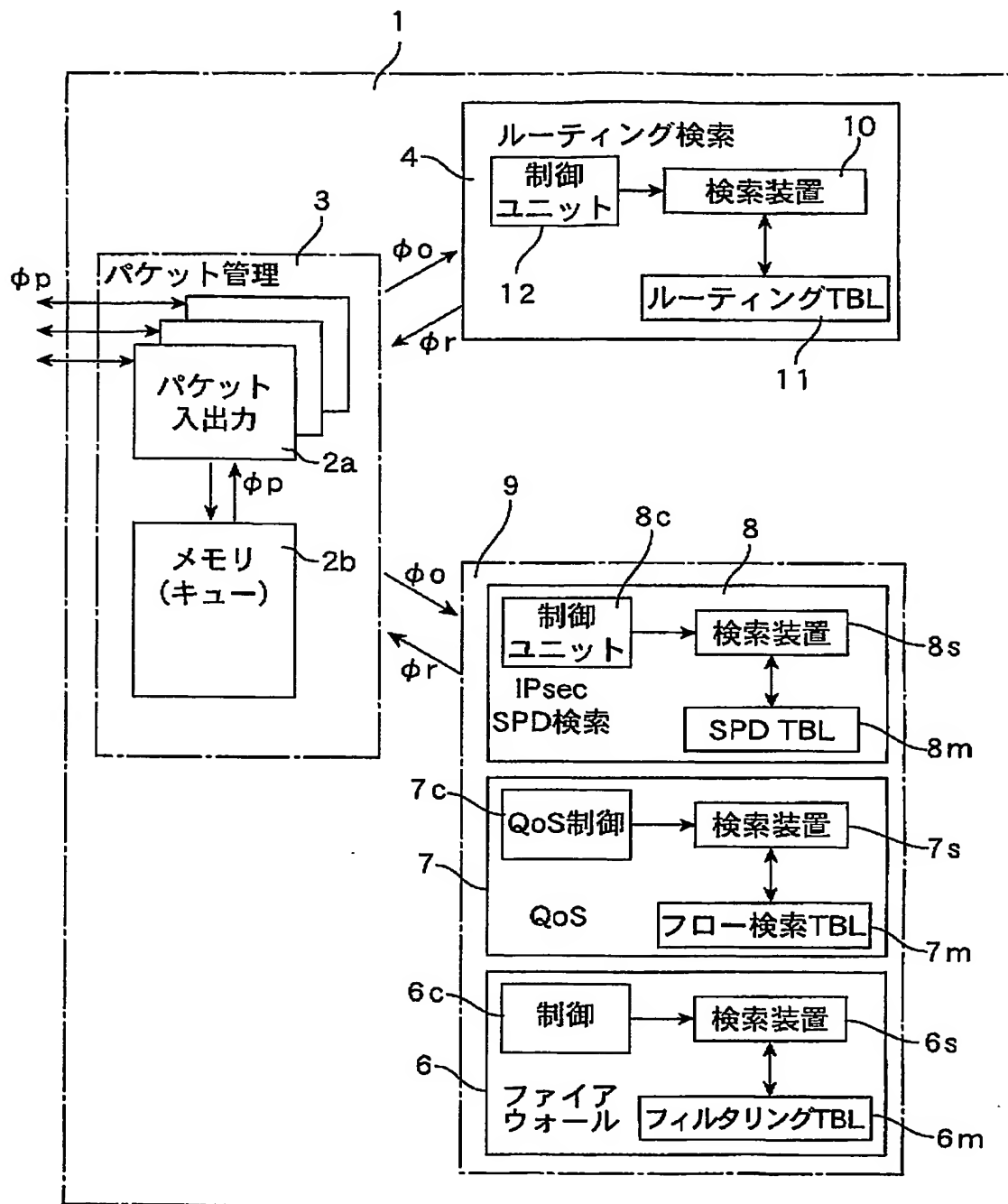
【図 72】 イベント連動型単純状態遷移を示す図である。

【符号の説明】

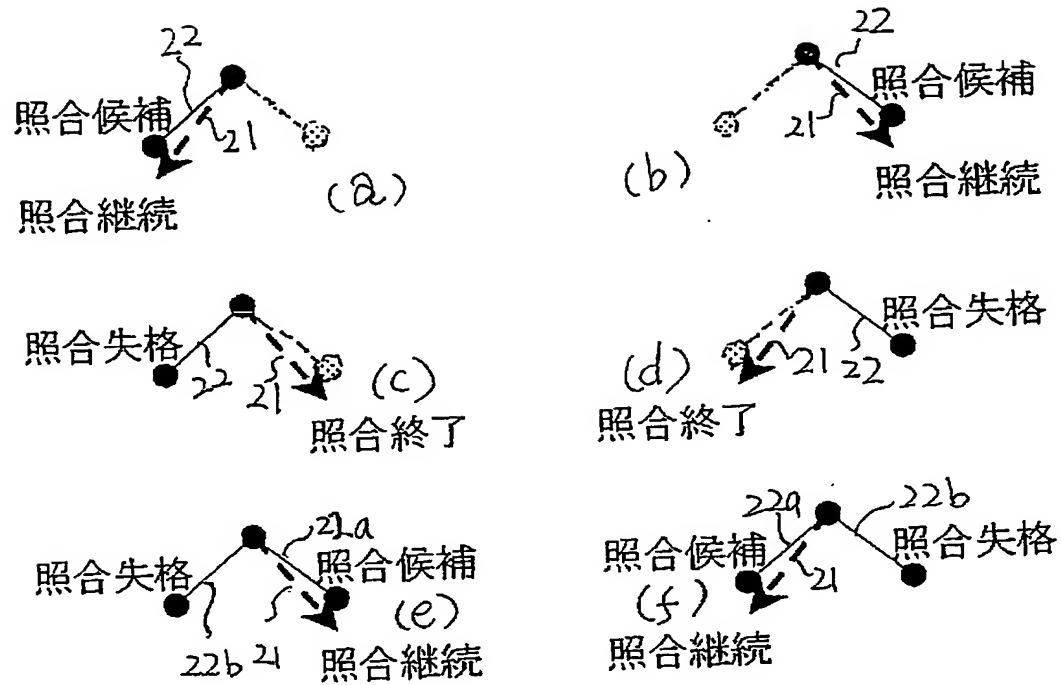
- 1 データ管理装置
- 10 検索装置
- 11 SDRAM (メモリ)
- 15 分類装置
- 50 照合装置

【書類名】 図面

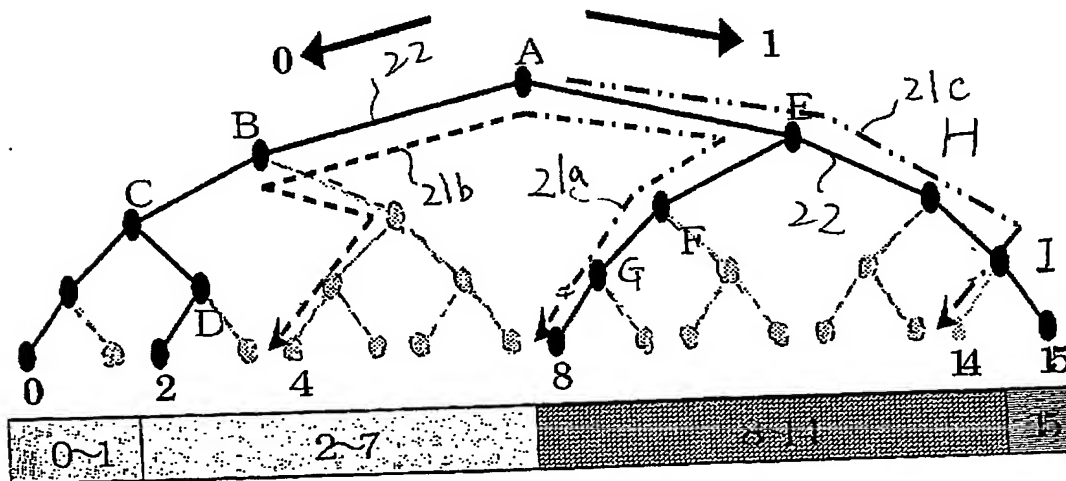
【図1】



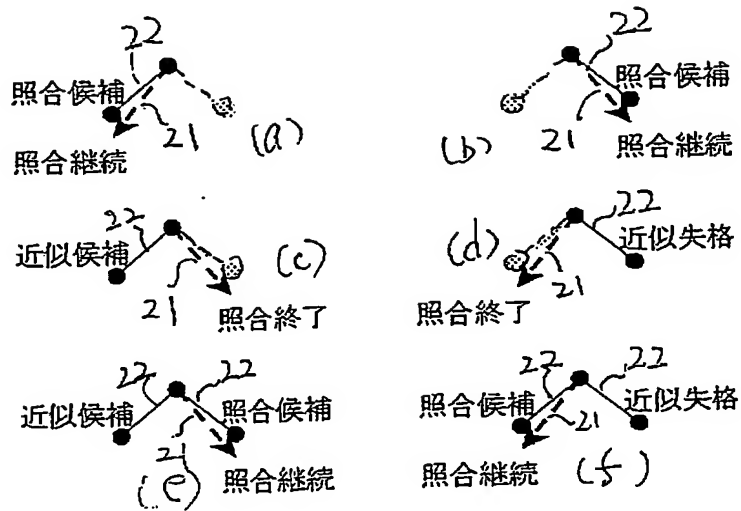
【図 2】



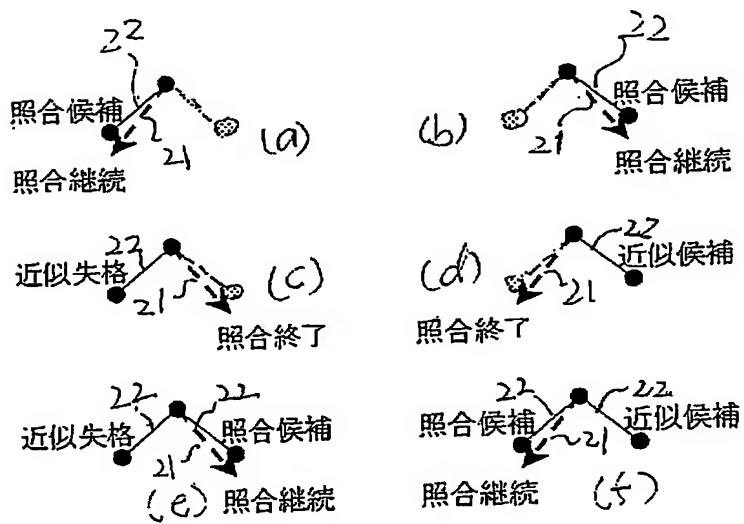
【図 3】



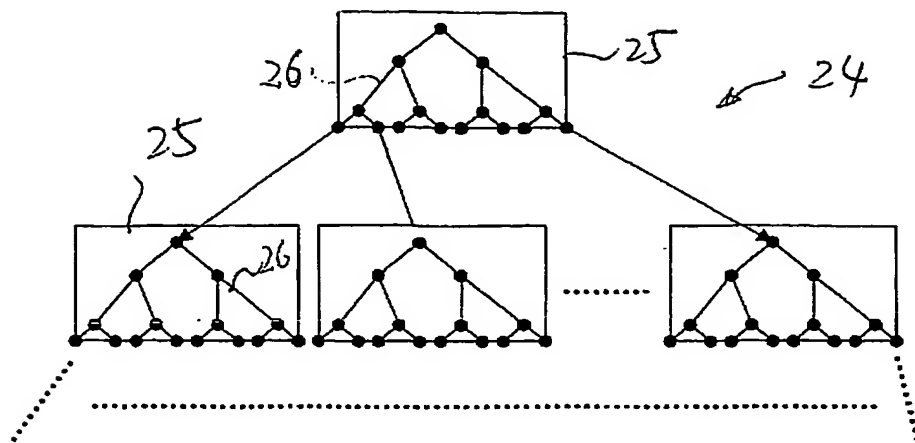
【図 4】



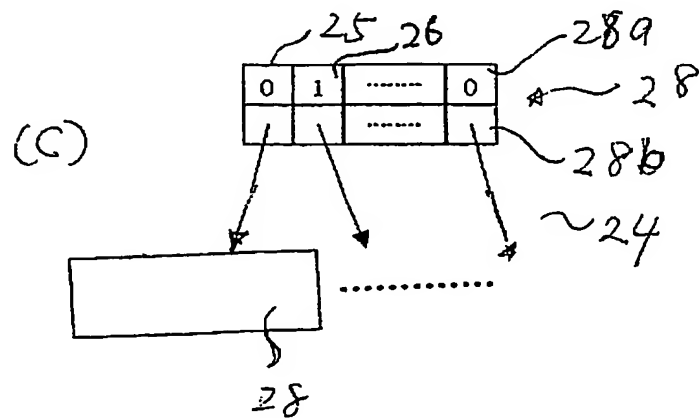
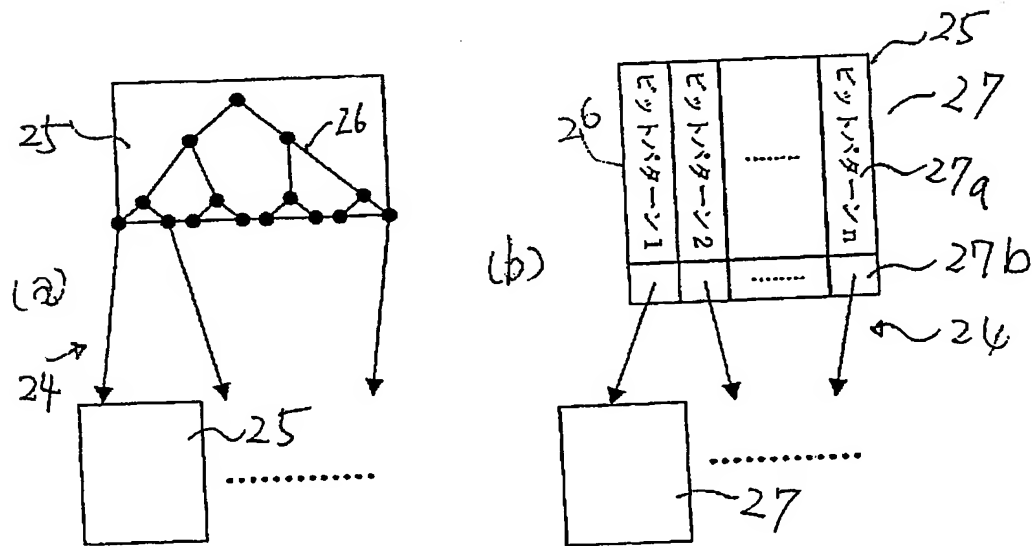
【図 5】



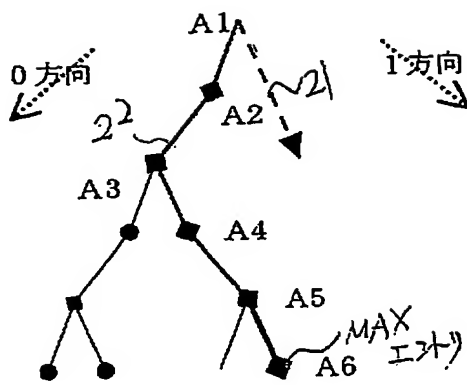
【図6】



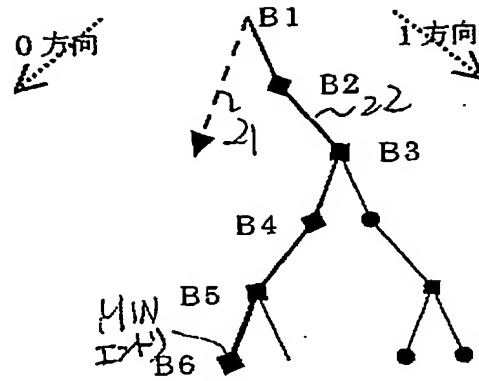
【図7】



【図8】

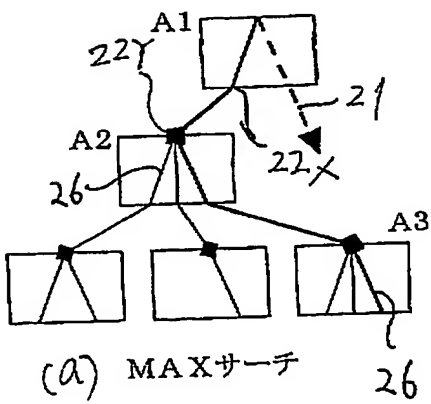


(a) MAXサーチ

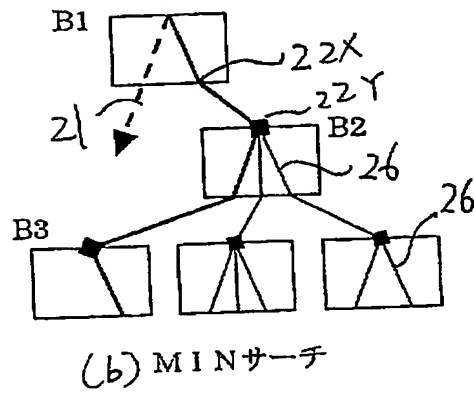


(b) MINサーチ

【図9】

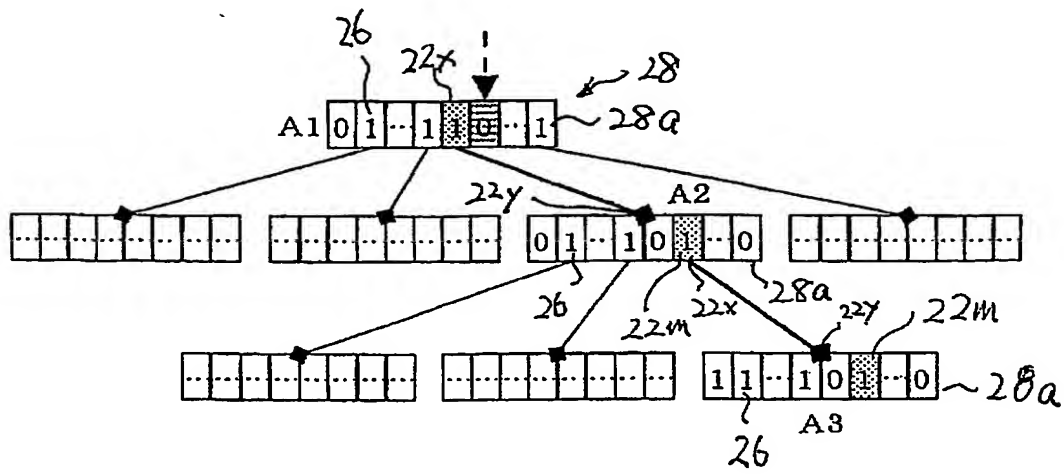


(a) MAXサーチ

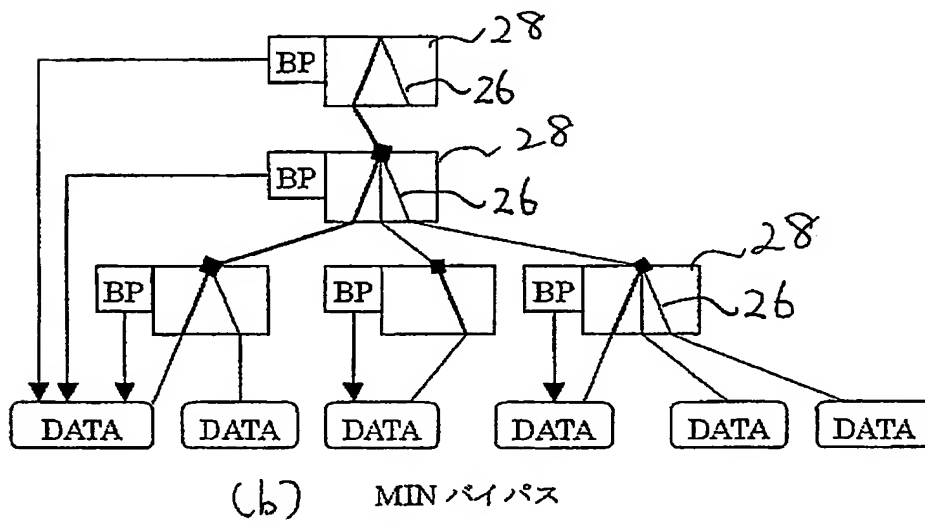
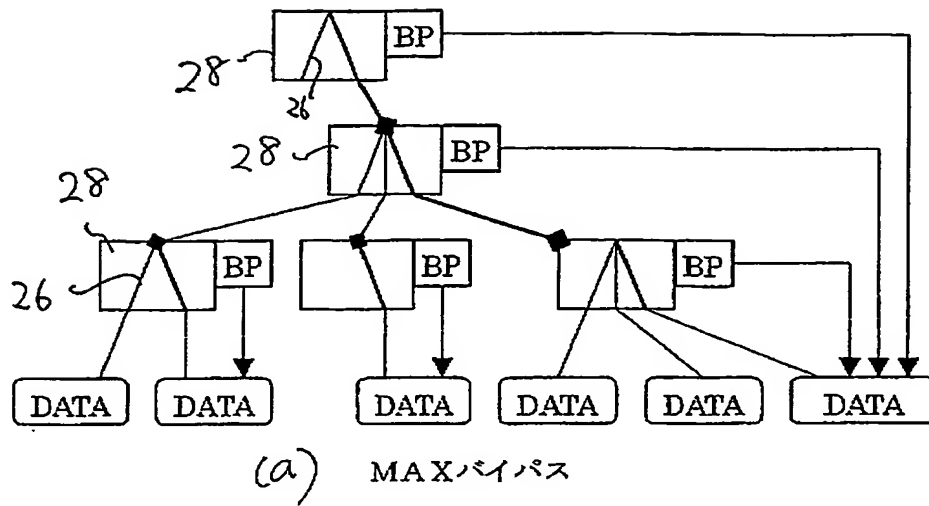


(b) MINサーチ

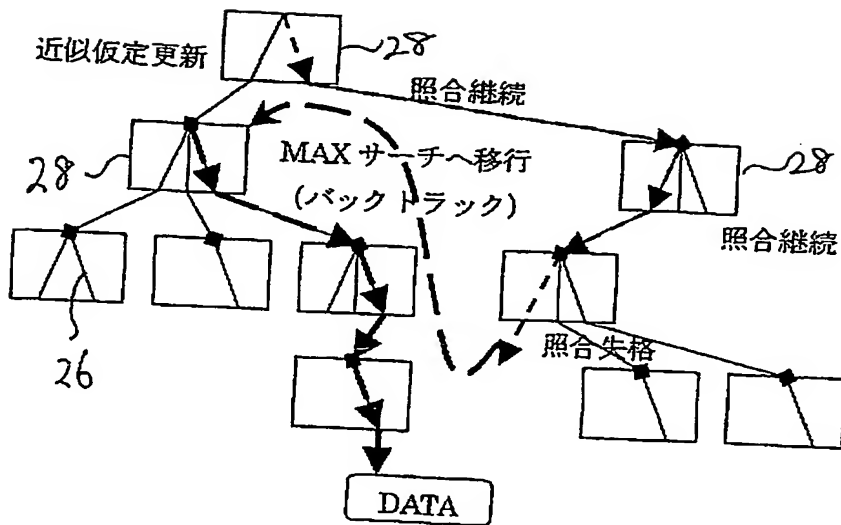
【図 10】



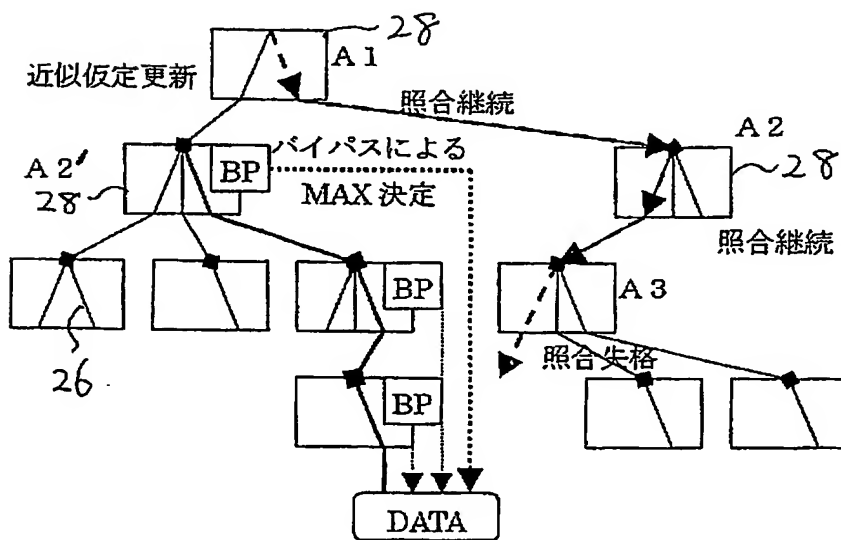
【図 11】



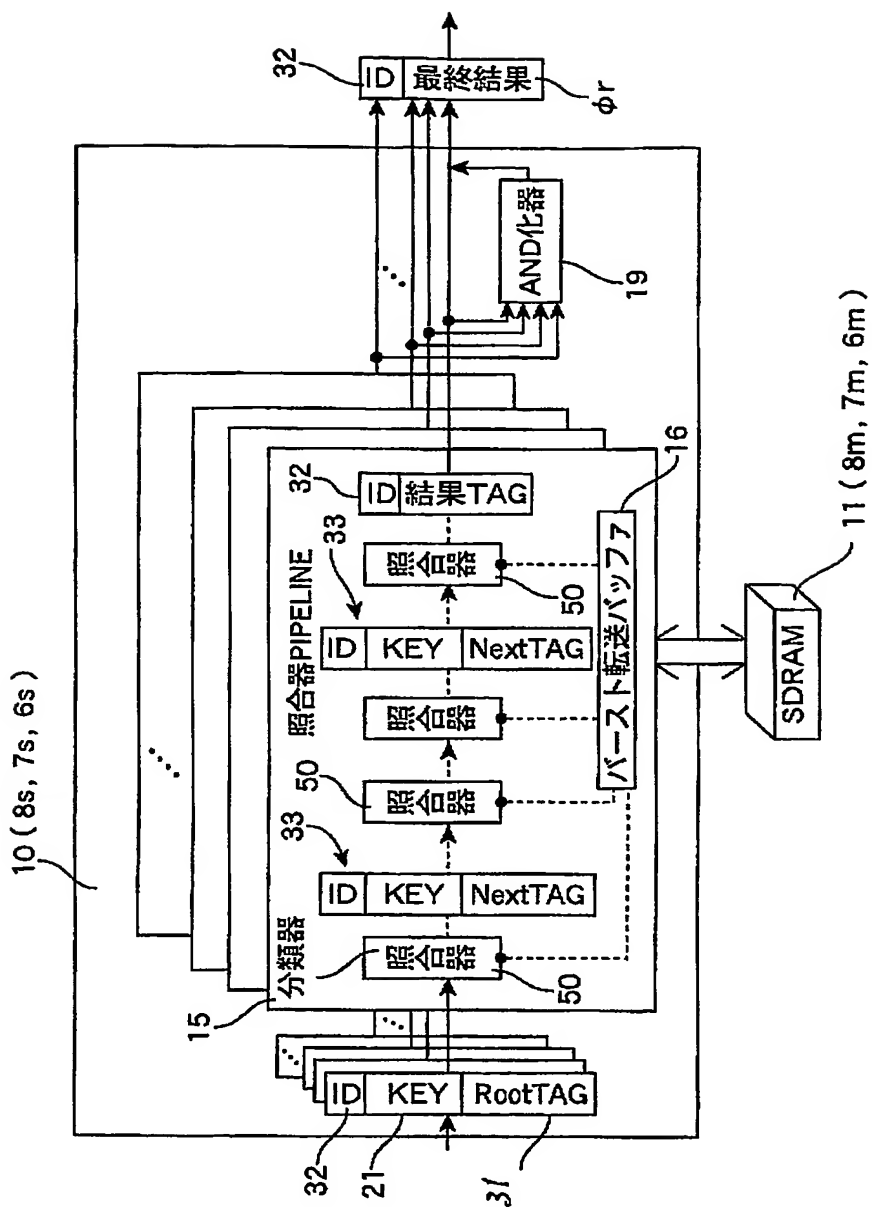
【図 12】



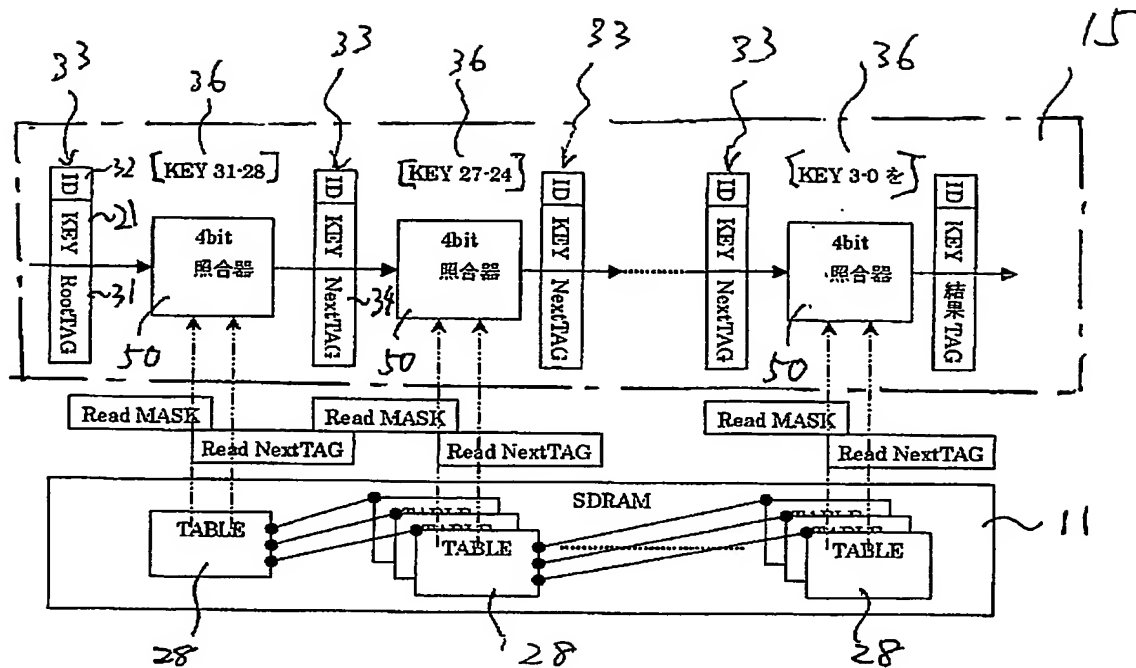
【図 13】



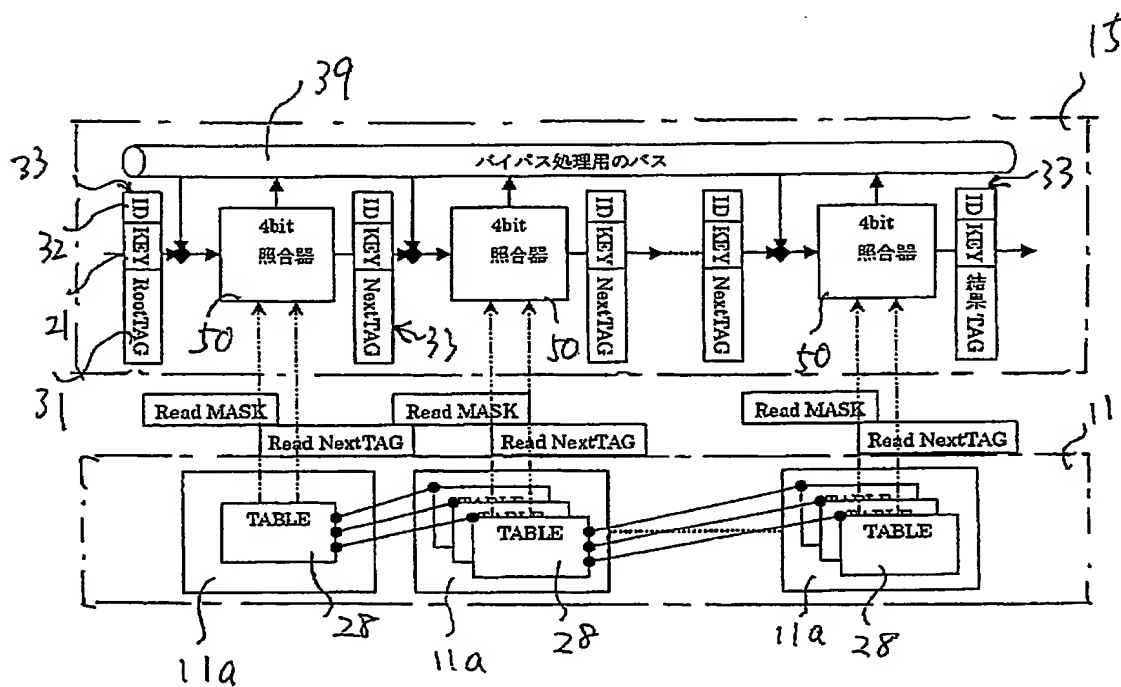
【図 14】



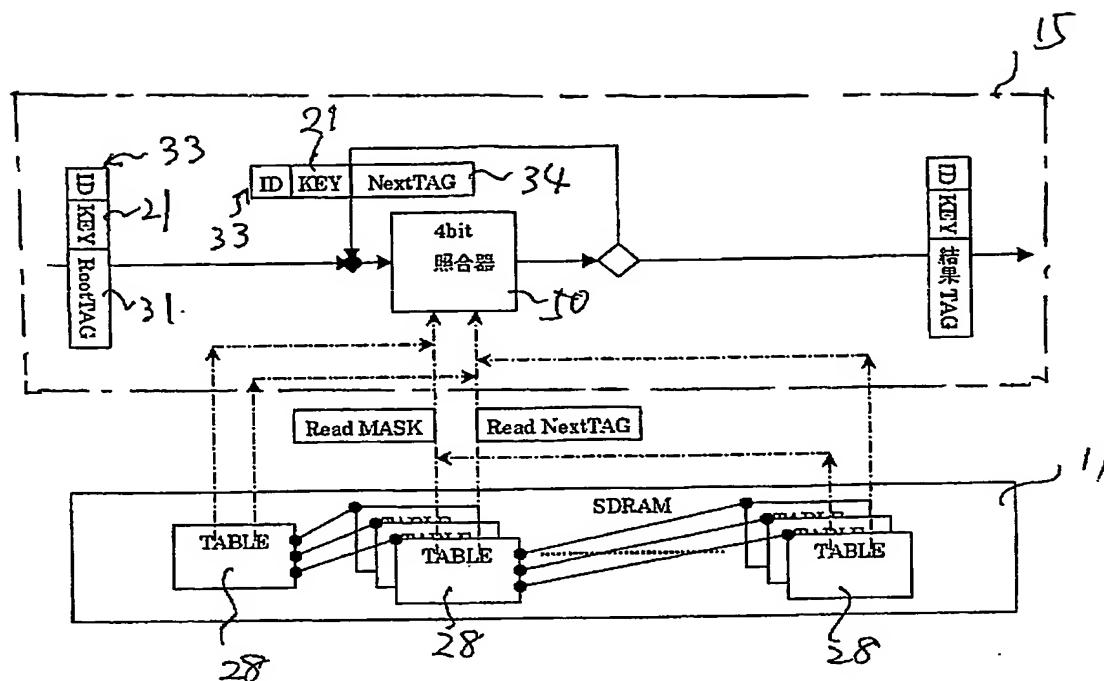
【図15】



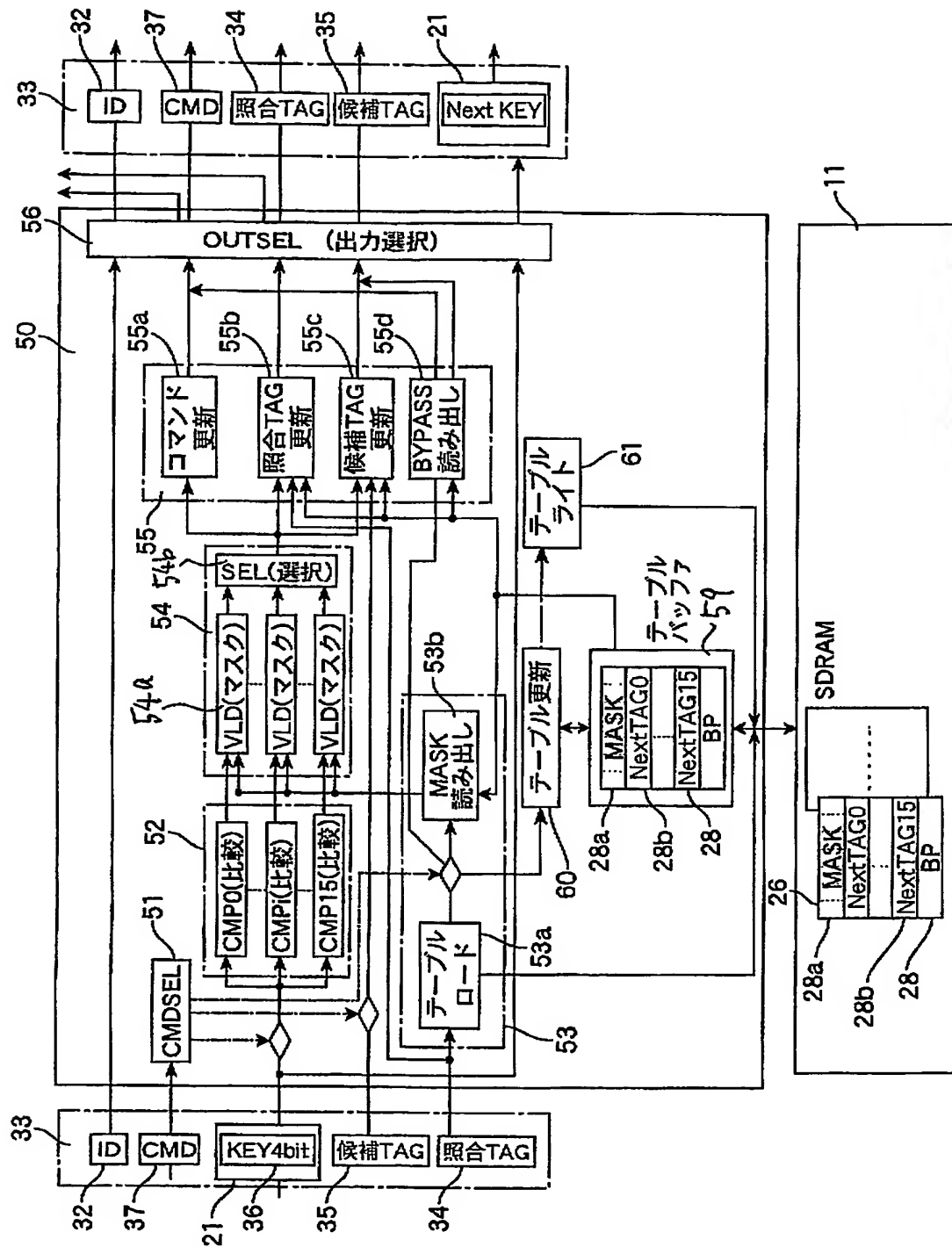
【図16】



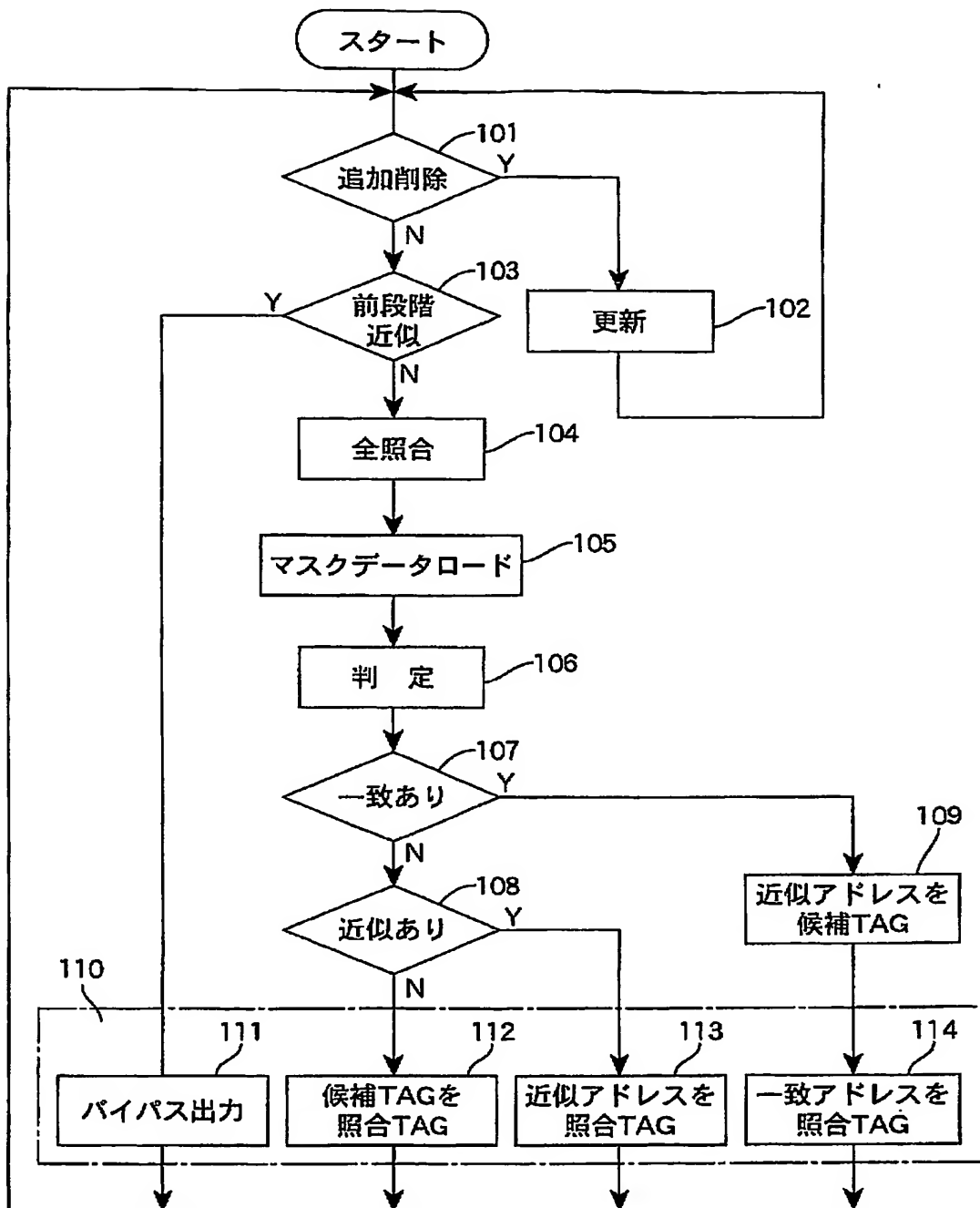
【図 17】



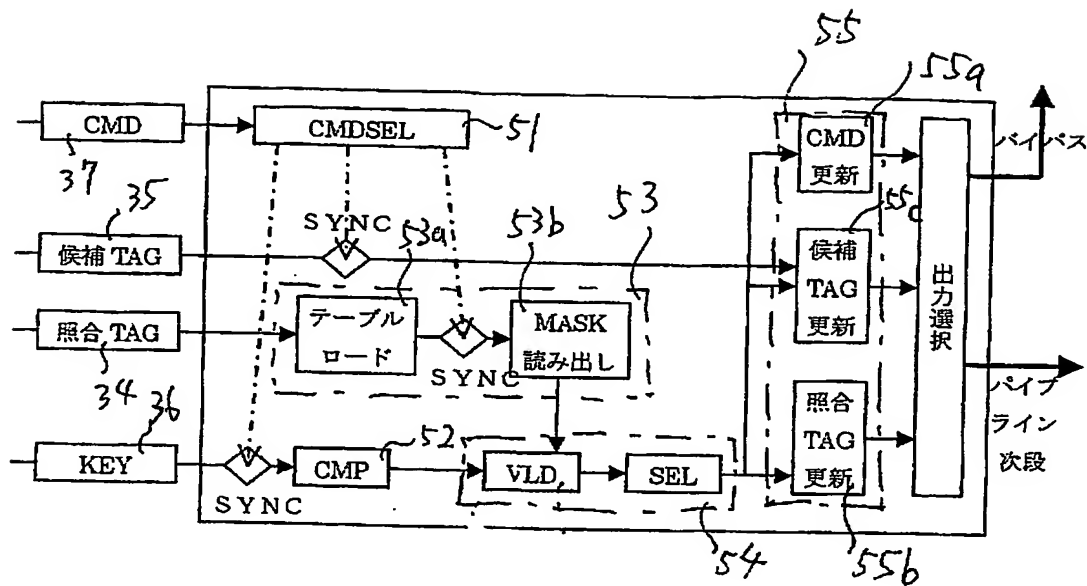
【圖 18】



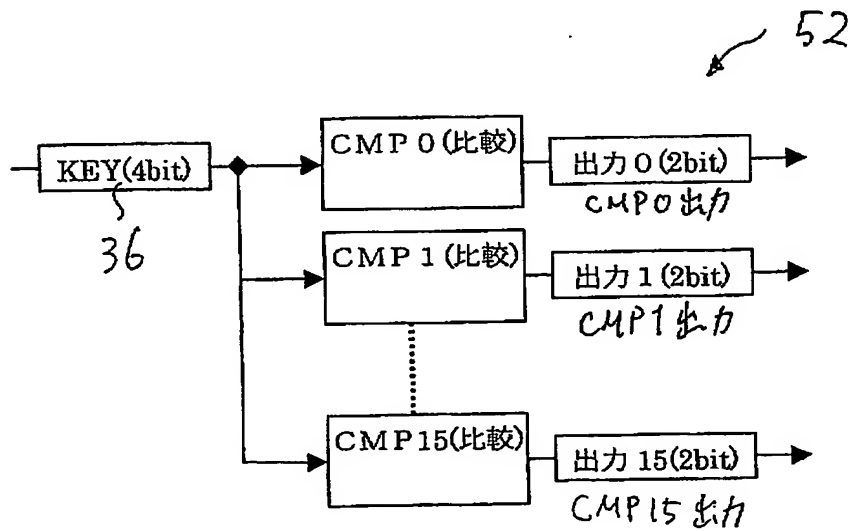
【図 19】



【図 20】



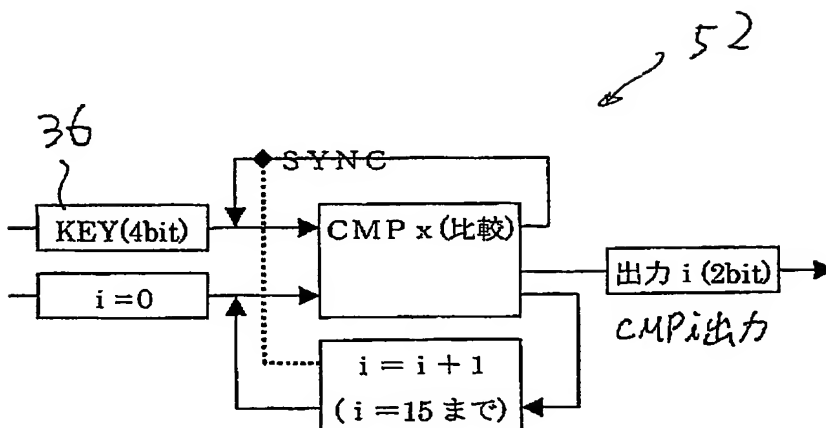
【図 21】



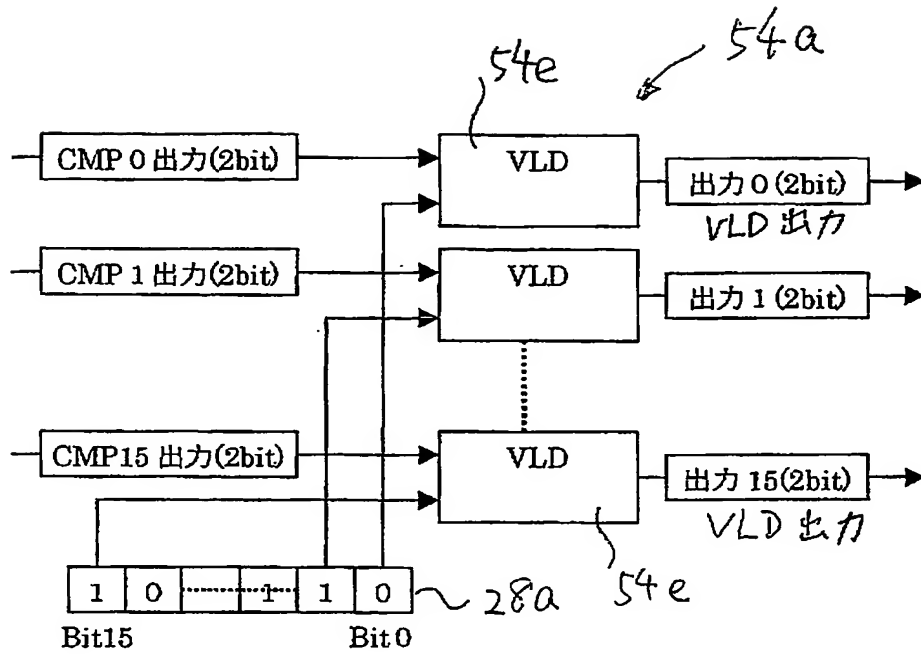
【図 2 2】

比較器CMP i の演算論理		
CMP i	入力 x	出力
	$x < i$	10 (「 $<$ 」)
	$x = i$	01 (「 $=$ 」)
	$x > i$	11 (「 $>$ 」)
$i = 0 \sim 15$ (固定値)		
$x = 0 \sim 15$ (4 bit 入力値)		

【図 2 3】



【図 2 4】



【図 2 5】

マスク器VLDの演算論理

CMP i 出力	MASK 値	VLD 出力
10	0	00 (「×」)
01	0	00 (「×」)
11	0	00 (「×」)
10	1	10 (「<」)
01	1	01 (「=」)
11	1	11 (「>」)
MASK 値による AND 演算		

【図 26】

完全一致検索の選択論理

VLD出力	出力 G_i
00 ($\Gamma \times J$)	0
10 ($\Gamma < J$)	0
01 ($\Gamma = J$)	1
11 ($\Gamma > J$)	0
$G_i(i) = \overline{\text{Bit1}} * \text{Bit0}$ $S_i = \sum G_i(i)$ $N_i = G_i^{-1}(1)$ ($G_i(i) = 1$ となる i を求める逆関数)	

【図 27】

範囲検索の照合継続選択論理 (右倒れ一致型)

VLD出力	中間 F_r	出力 G_r
00 ($\neg \times$)	0	0
10 ($\neg <$)	0	0
01 ($\neg =$)	0	0
11 ($\neg >$)	1	$G_r(i)$

$F_r(i) = \text{Bit } 1 * \text{Bit } 0$
 $H_r(-1) = 0$ (漸化式仮想初期値)
 $H_r(i) = H_r(i-1) + F_r(i)$
 $G_r(i) = \overline{H_r(i-1)} * H_r(i)$
 $S_r = \Sigma G_r(i)$
 $N_r = G_r^{-1}(1)$
 $(G_r(i) = 1 \text{ となる } i \text{ を求める逆関数})$

$H_r(i-1)$	$F_r(i)$	$H_r(i)$
0	0	0
0	1	1
1	0	1
1	1	1

$H_r(i-1)$	$H_r(i)$	$G_r(i)$
0	0	0
0	1	1
1	0	0
1	1	0

【図 28】

範囲検索の選択論理 (左倒れ一致)

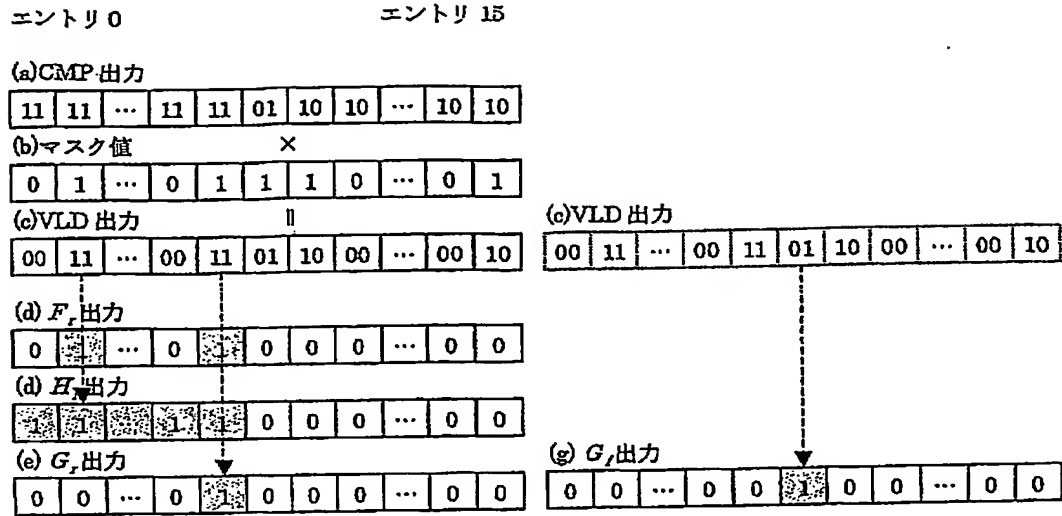
VLD出力	中間 F_i	出力 G_i
00 ($\neg \times$)	0	0
10 ($\neg <$)	1	$G_i(i)$
01 ($\neg =$)	0	0
11 ($\neg >$)	0	0

$F_i(i) = \text{Bit } 1 * \overline{\text{Bit } 0}$
 $H_i(16) = 0$ (漸化式仮想初期値)
 $H_i(i) = H_i(i+1) + F_i(i)$
 $G_i(i) = \overline{H_i(i+1)} * H_i(i)$
 $S_i = \sum G_i(i)$
 $N_i = G_i^{-1}(1)$
 $(G_i(i) = 1 \text{ となる } i \text{ を求める逆関数})$

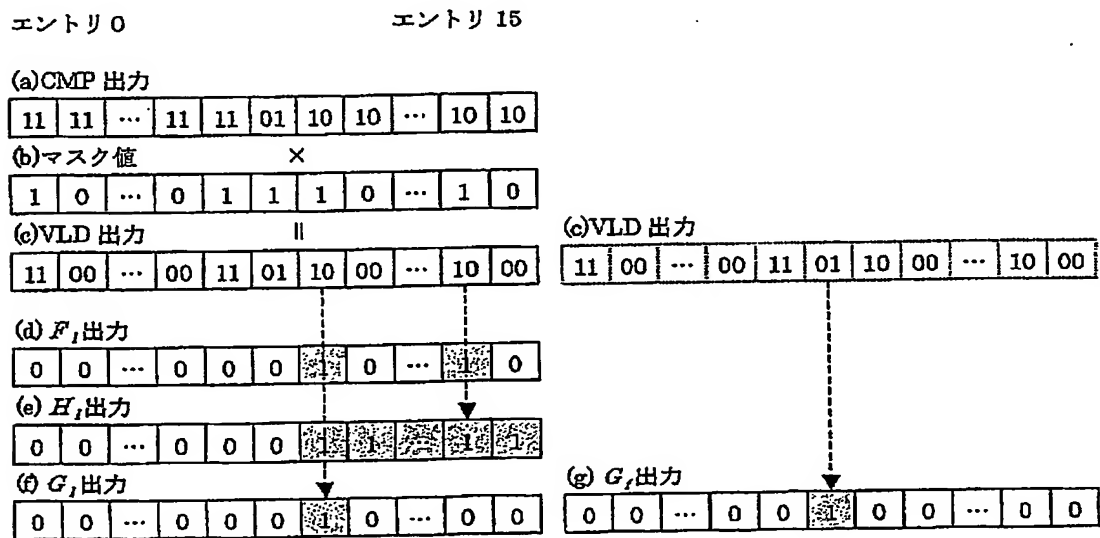
$H_i(i+1)$	$F_i(i)$	$H_i(i)$
0	0	0
0	1	1
1	0	1
1	1	1

$H_i(i+1)$	$H_i(i)$	$G_i(i)$
0	0	0
0	1	1
1	0	0
1	1	0

【図 29】



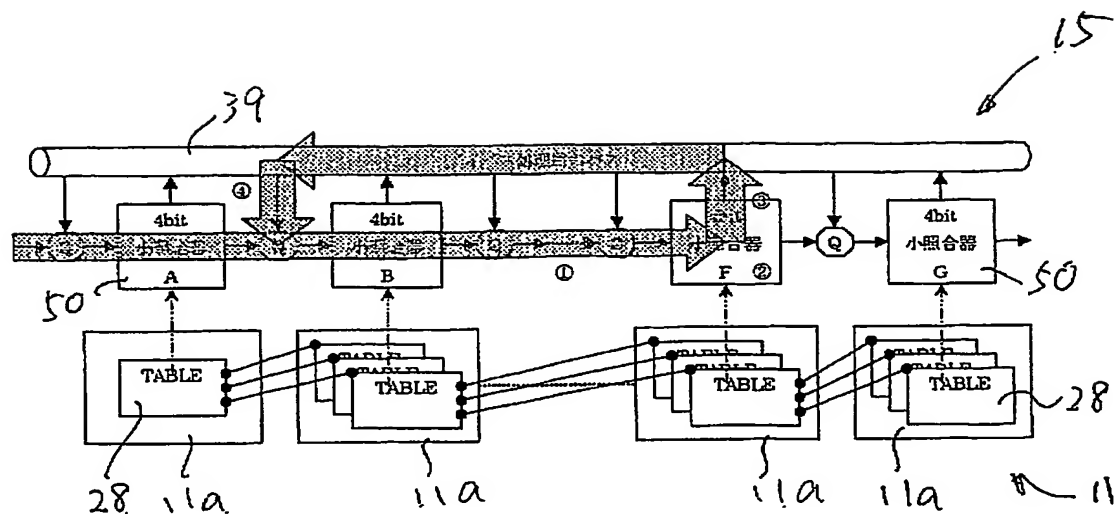
【図 30】



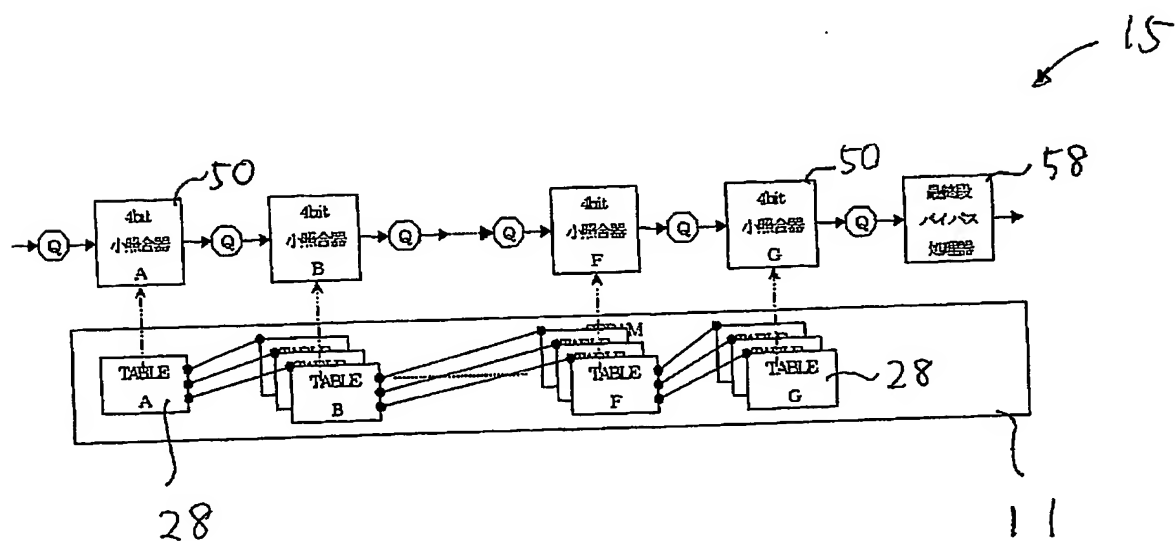
【図 31】

種別	コマンド	説明
完全一致検索	FMATCHEXEC	照合継続
	FMATCHHIT	ヒット
	FMATCHFAIL	ミスヒット
範囲検索	MATCHEXEC	照合継続
	MATCHRDBP	バイパス読み出し
	MATCHFHIT	完全一致ヒット
	MATCHNHIT	近似ヒット
	MATCHFAIL	ミスヒット
登録	ADD	上位→下位
	ADDNEW	新規テーブル
	ADDFIN	登録完了
	ADDFAIL	登録失敗
	ADDUPDT	BP 更新フィード
	(ADDWCLR)	(待ち取り消し)
削除	DEL	下位継続
	DELFIN	削除完了
	DELFAIL	削除失敗
	DELUPDT	BP 更新フィード
	DELRDBP	下位 BP 参照
	DELWRBP	自 BP 更新

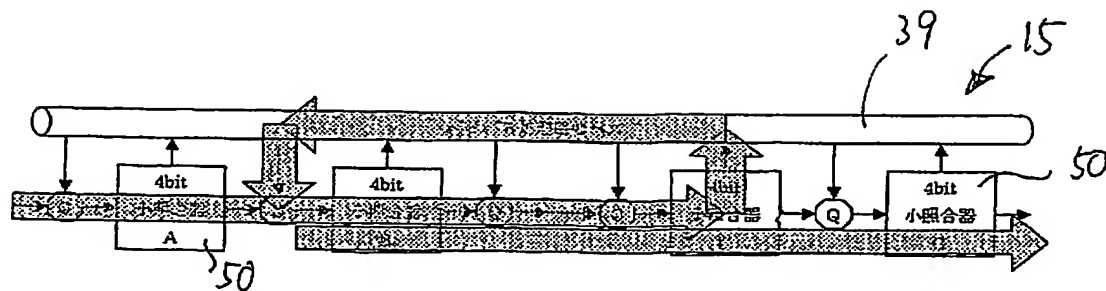
【図32】



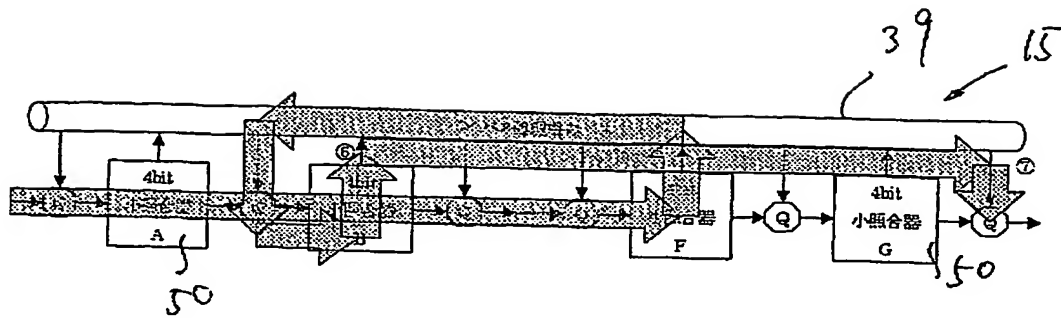
【図33】



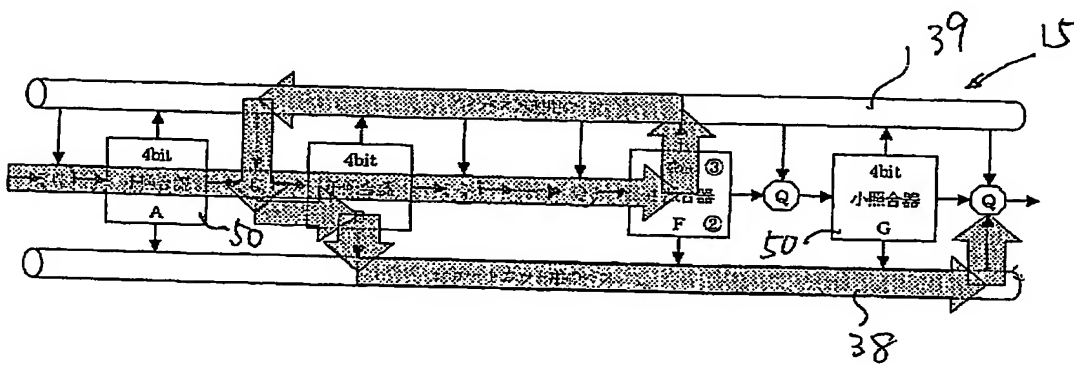
【図34】



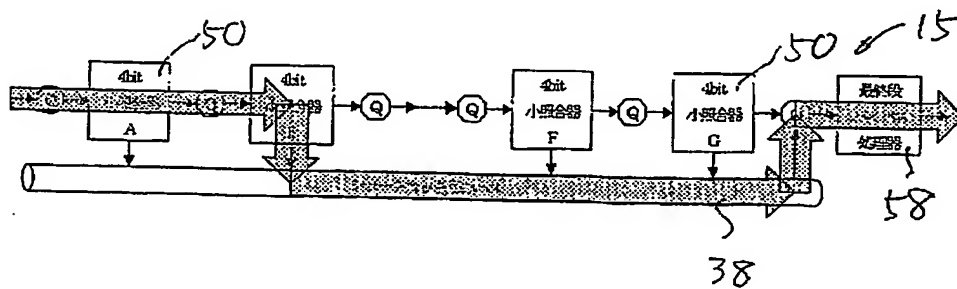
【図 35】



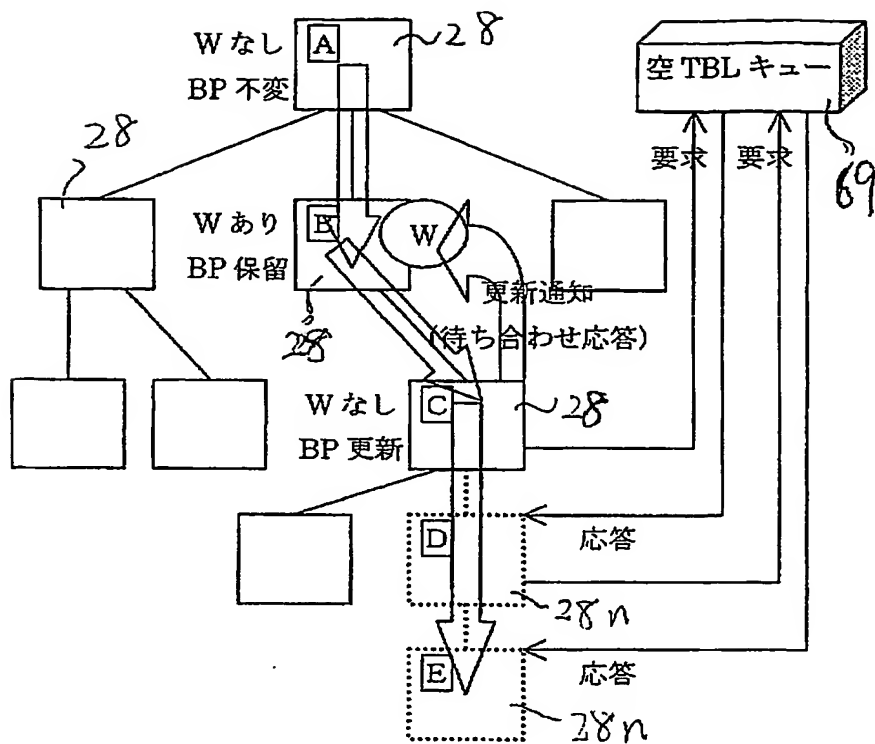
【図 36】



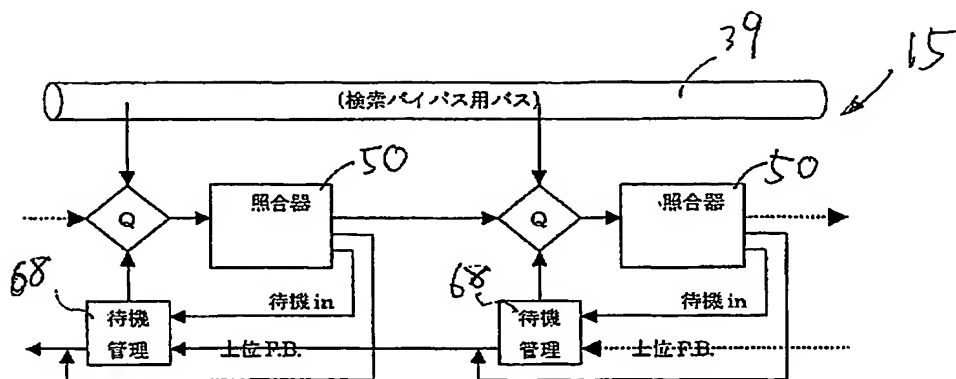
【図 37】



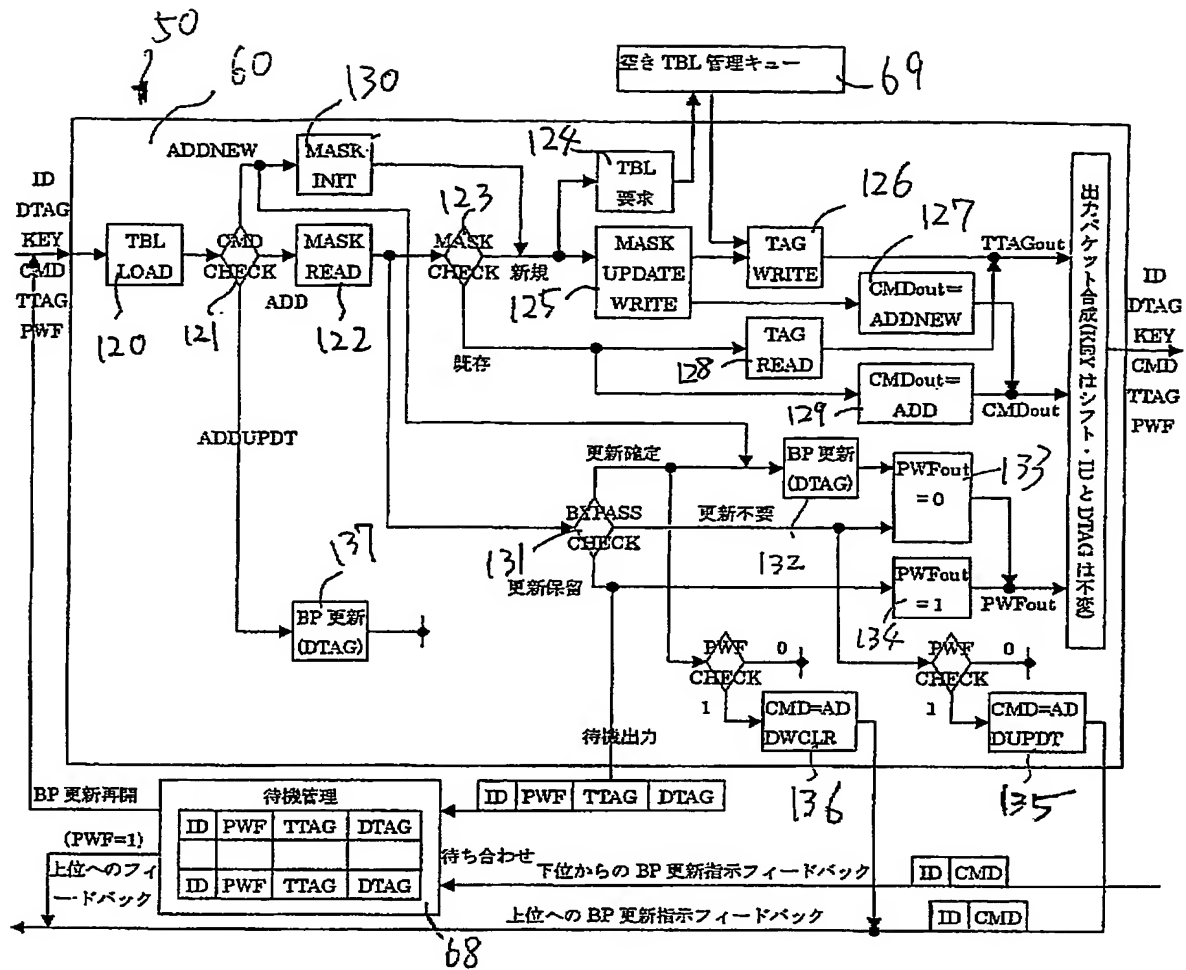
【図 38】



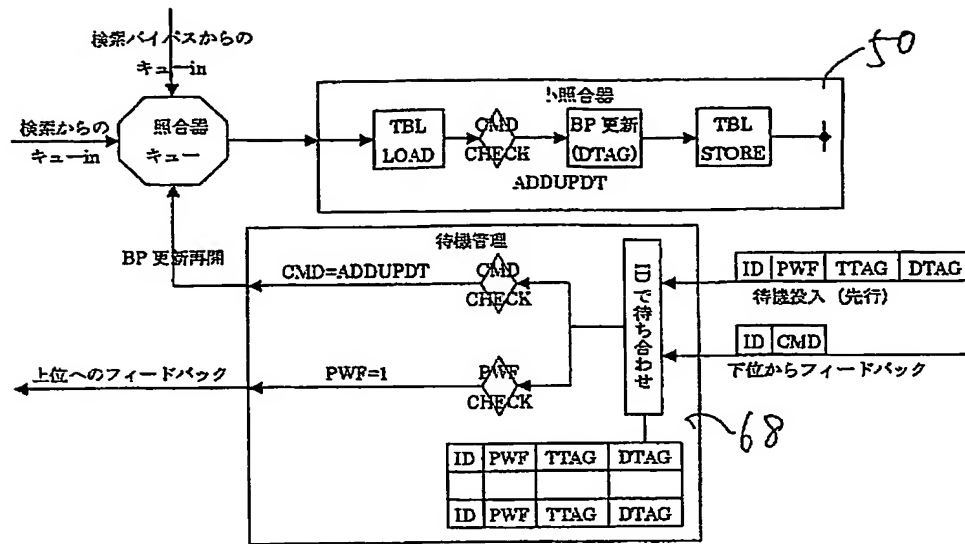
【図 39】



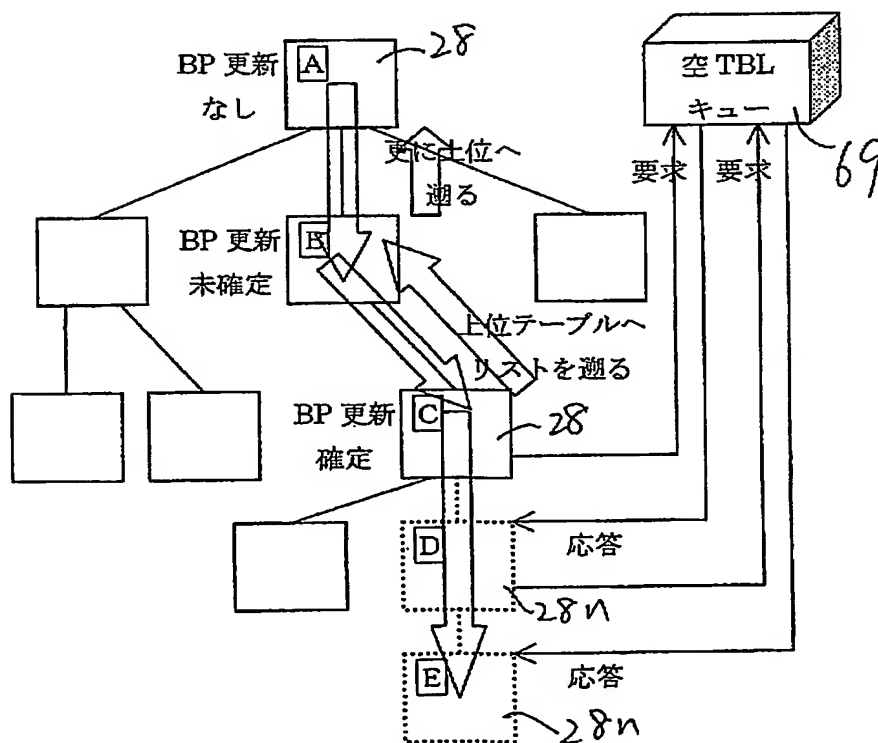
【図 40】



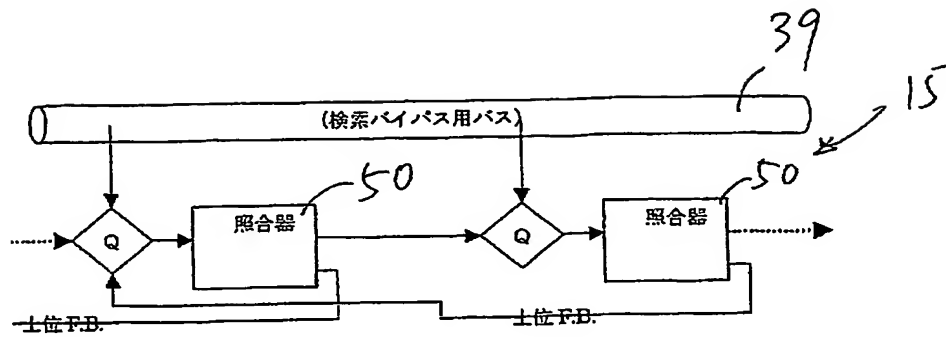
【図 4 1】



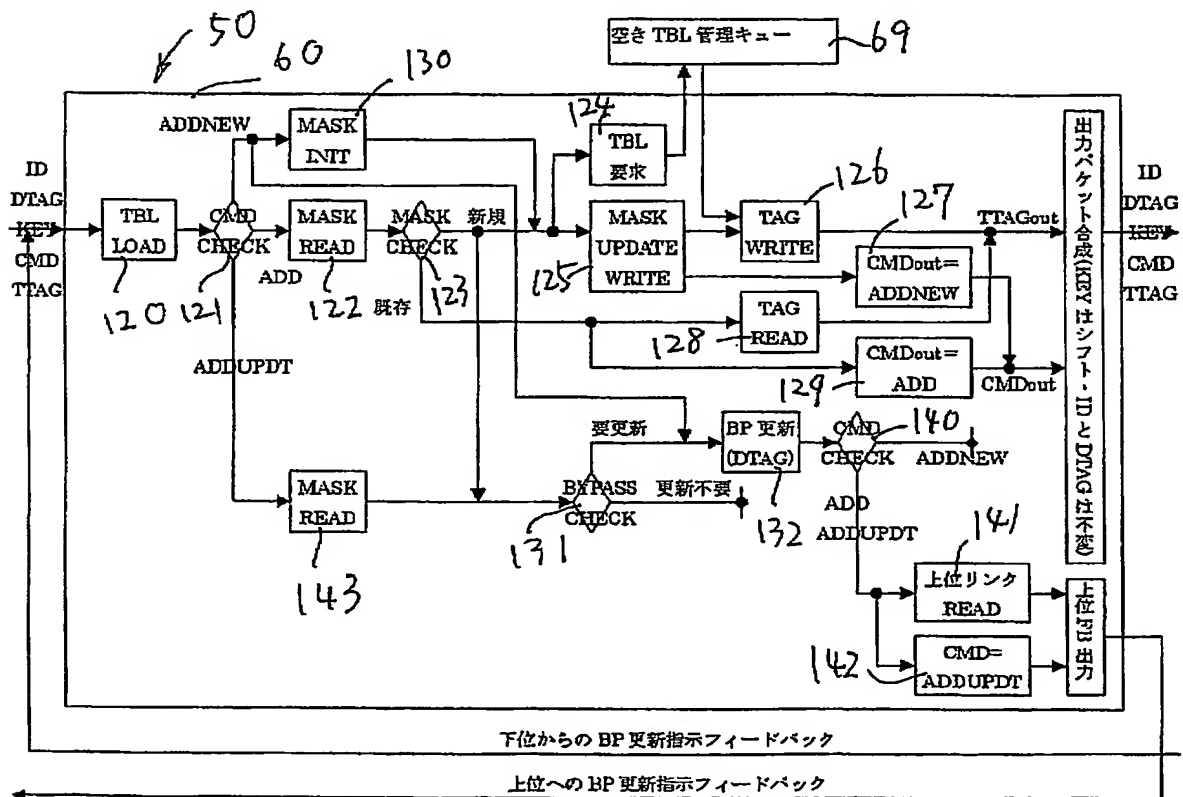
【図 4 2】



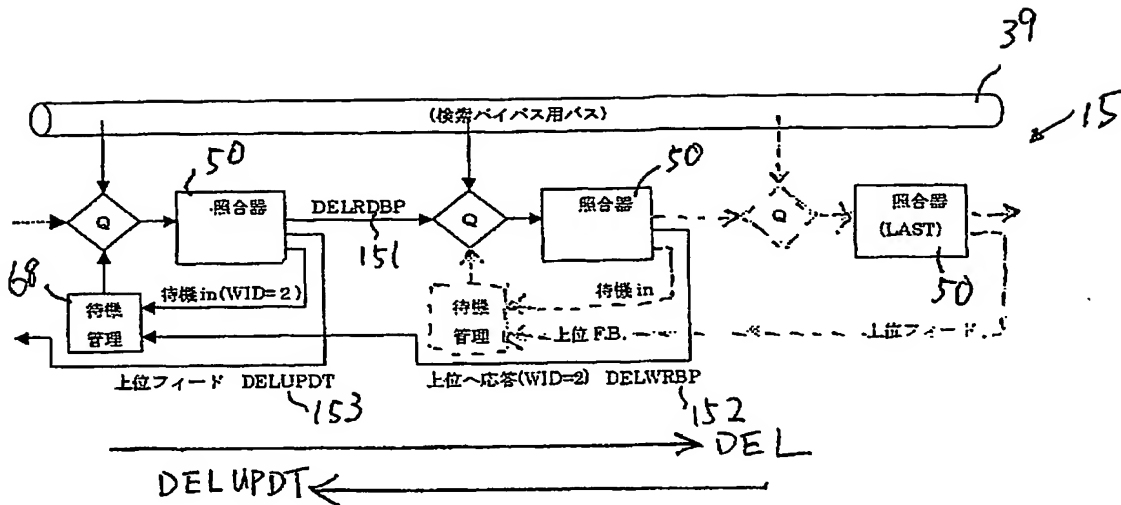
【図 4 3】



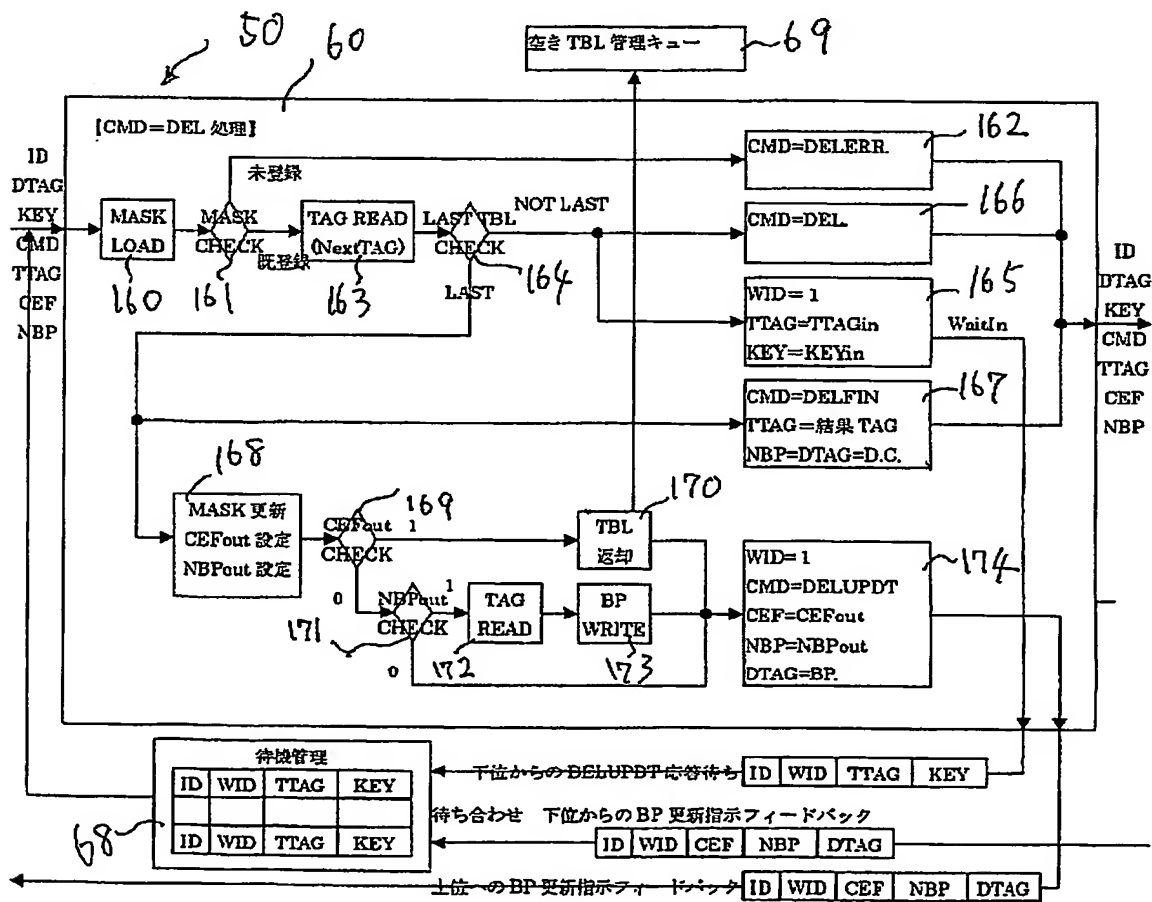
【図 4 4】



【図45】

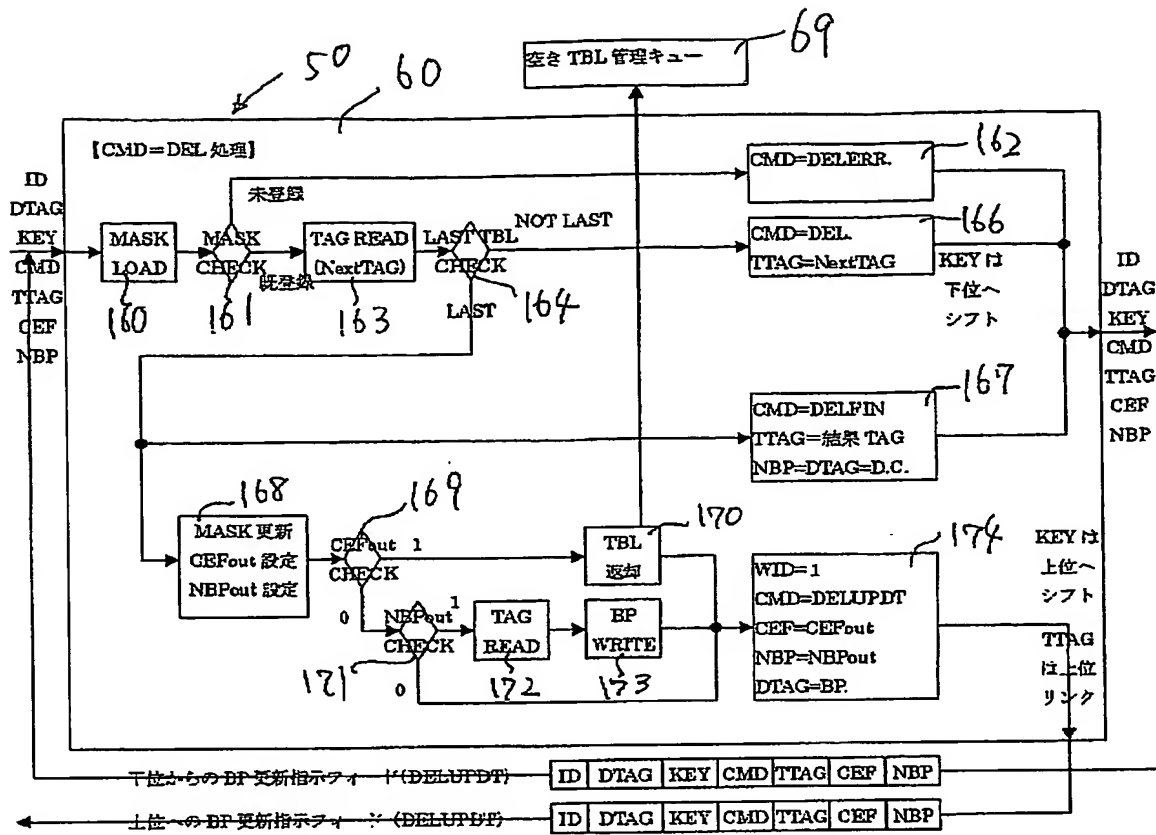


【図46】

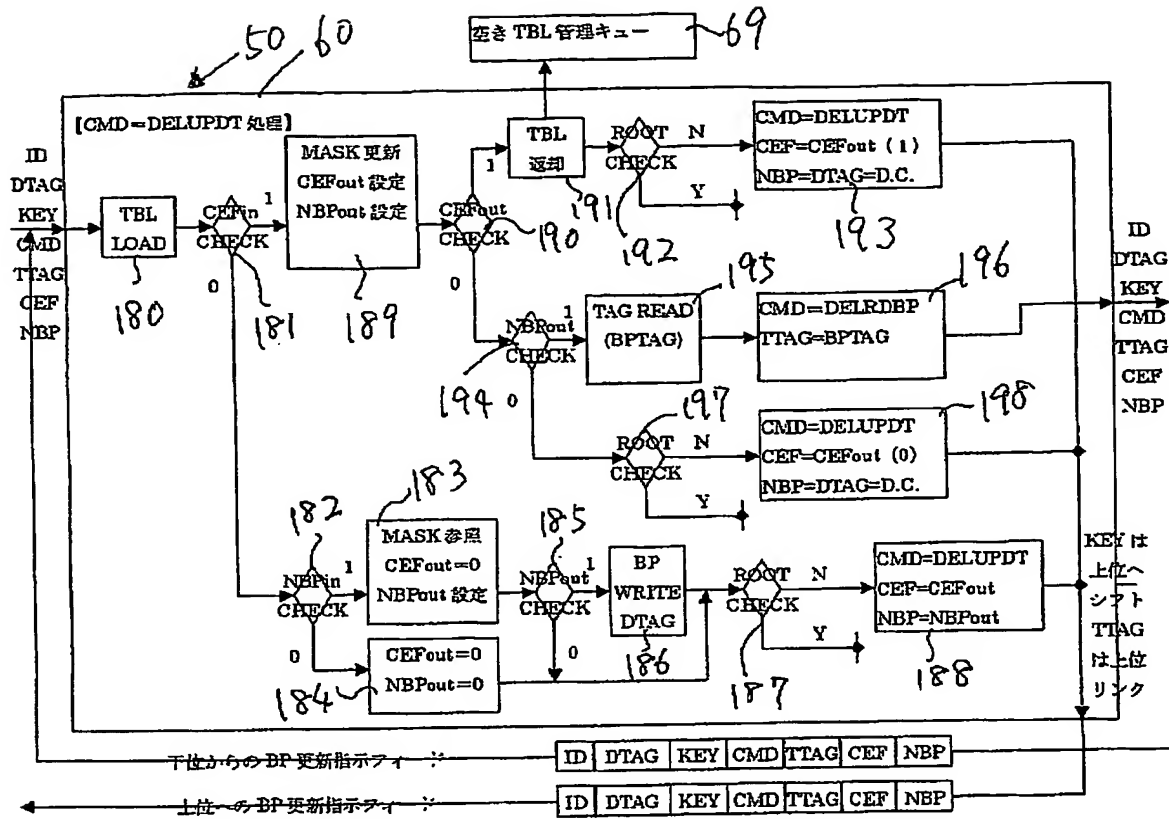


[illegible][illegible]

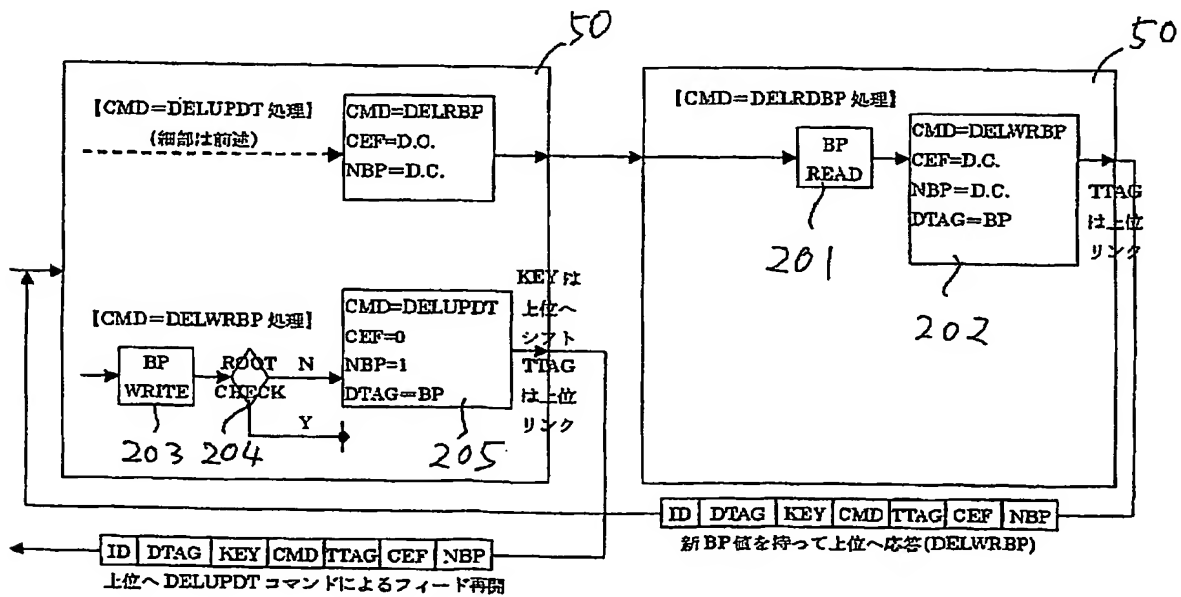
【図 49】



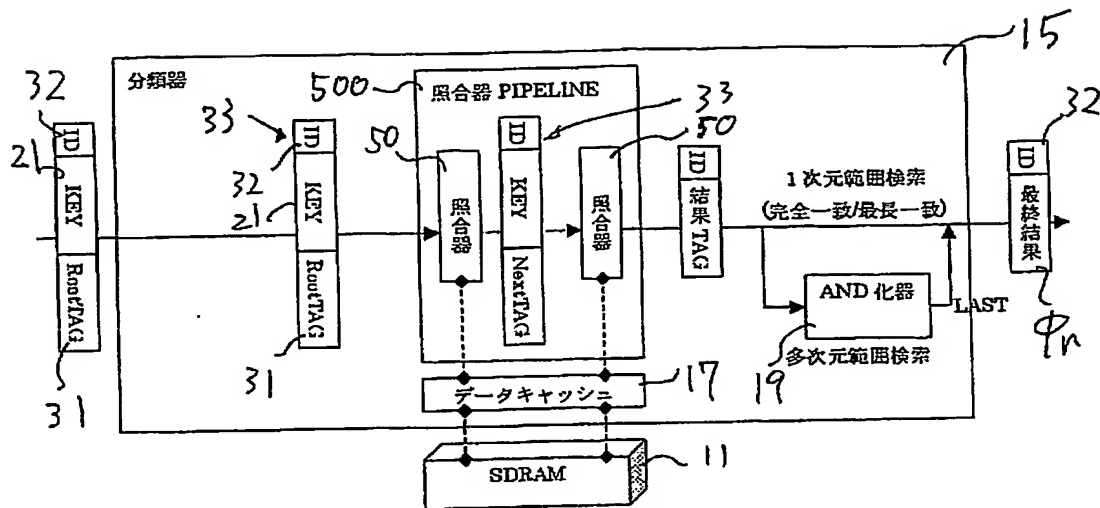
【図 50】



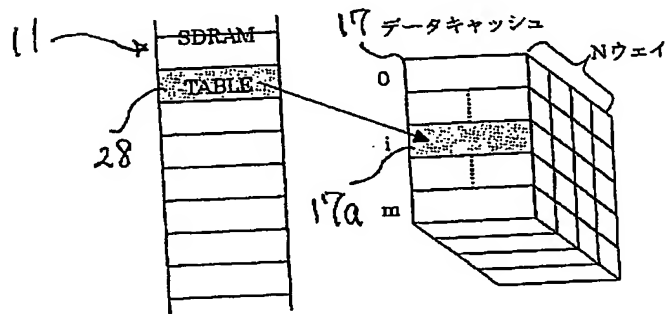
【図 51】



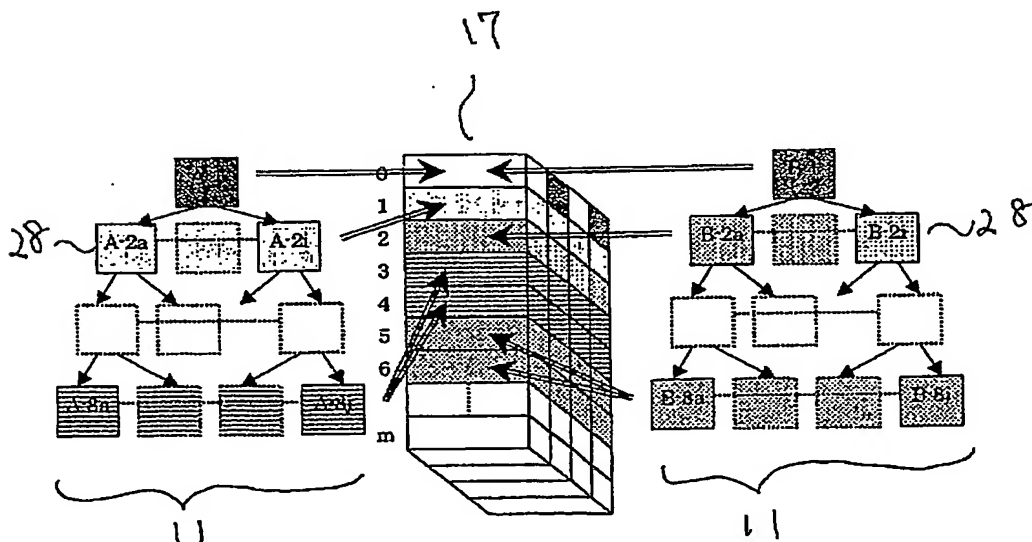
【図 5 2】



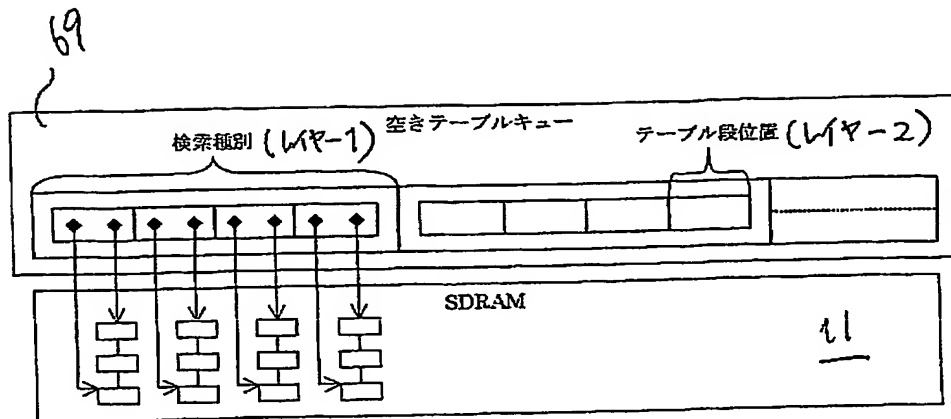
【図 5 3】



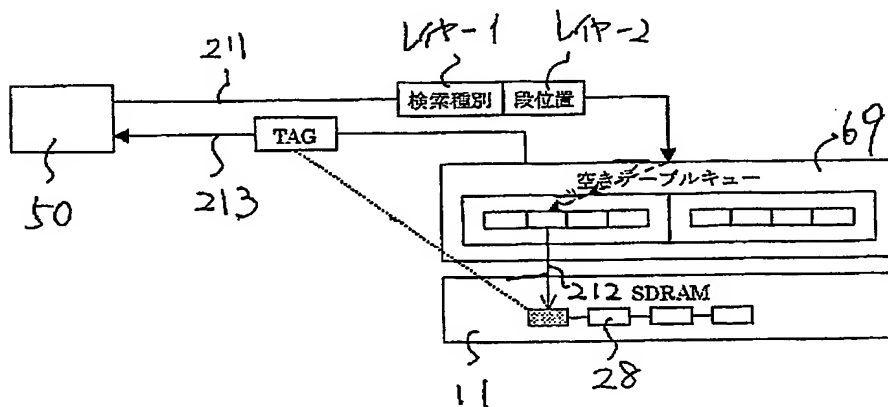
【図 5 4】



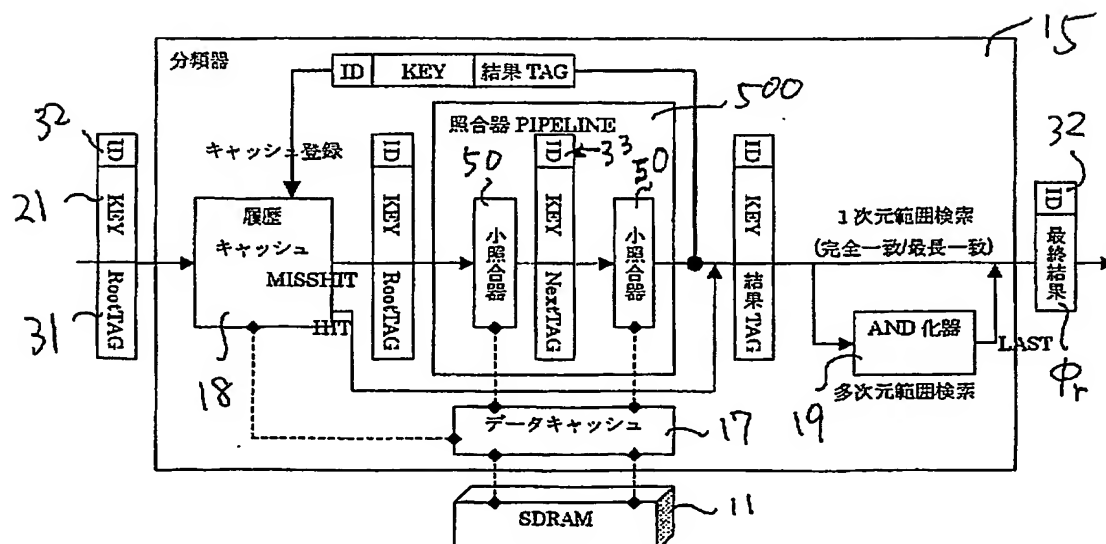
【図 5 5】



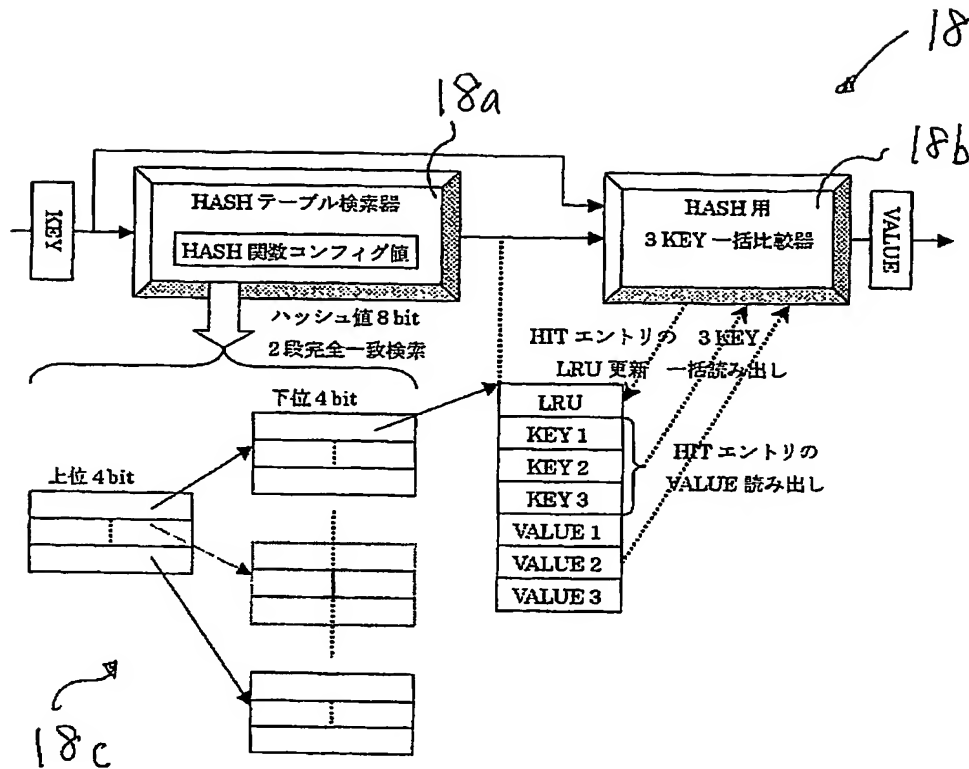
【図 5 6】



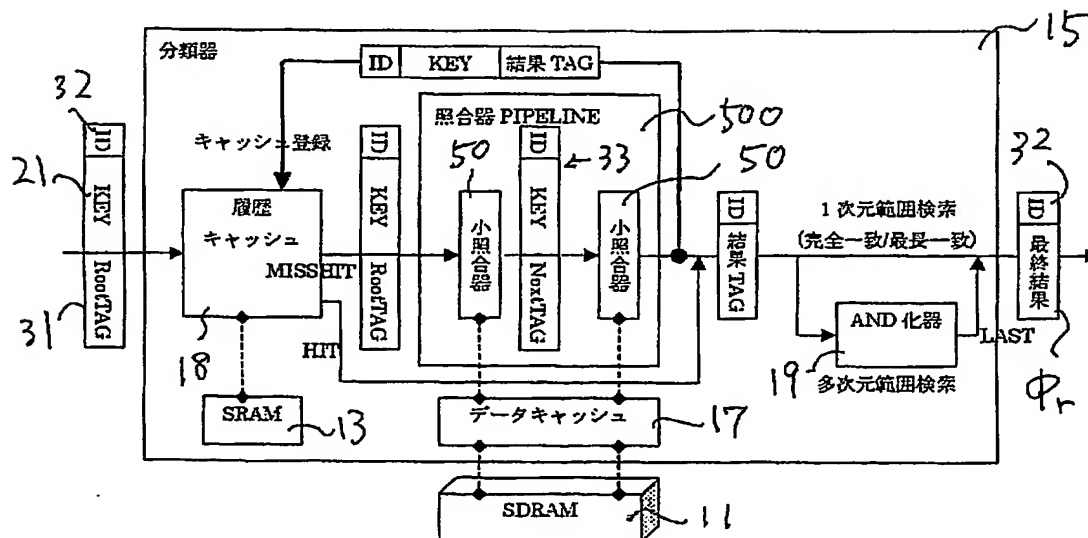
【図 5 7】



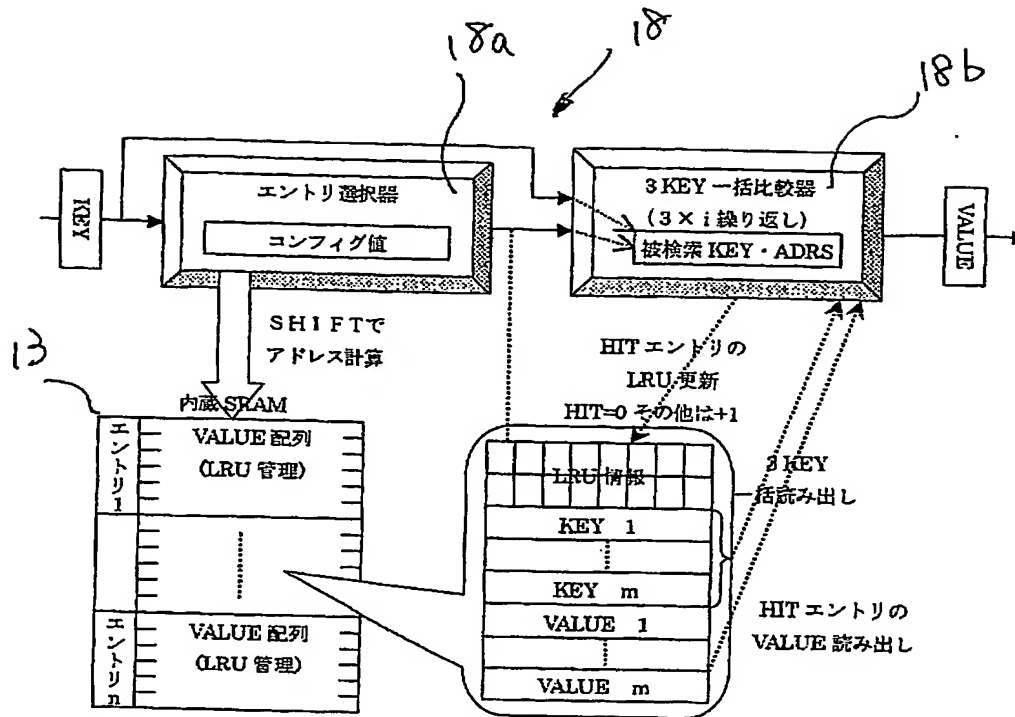
【図 58】



【図 5 9】



【図 60】

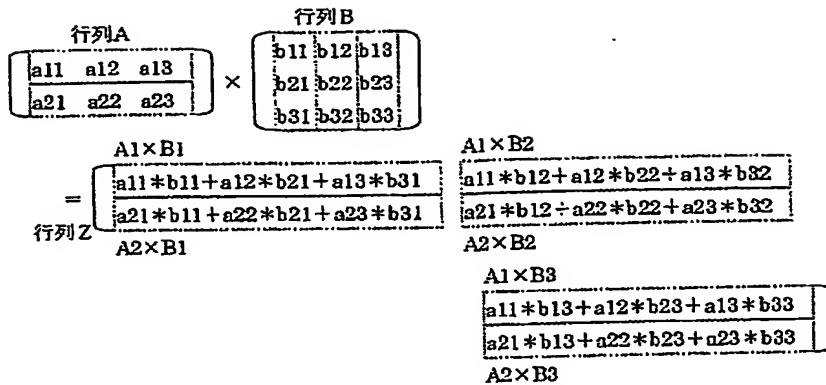


【図 61】

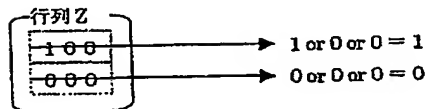
$$A_i \times B_j = \sum_k a_{ik} * b_{kj}$$

- Σ 内はブール乗算 (*) とブール加算 (+) に展開される。
- ここでのブール乗算・ブール加算の定義は、ブール乗算「*」は ENOR (Exclusive NOR、非排他論理和)、ブール加算「+」は AND である
 (ブール加算) $1+1=1$ $1+0=0$ $0+1=0$ $0+0=0$
 (ブール乗算) $1*1=1$ $1*0=0$ $0*1=0$ $0*0=1$
- ENOR は比較回路の=のロジックと同じであるので、 $A_i * B_j$ は比較等号回路を AND で結んだものだとして理解してもかまわない。行列で理解したほうが見通しが良い。

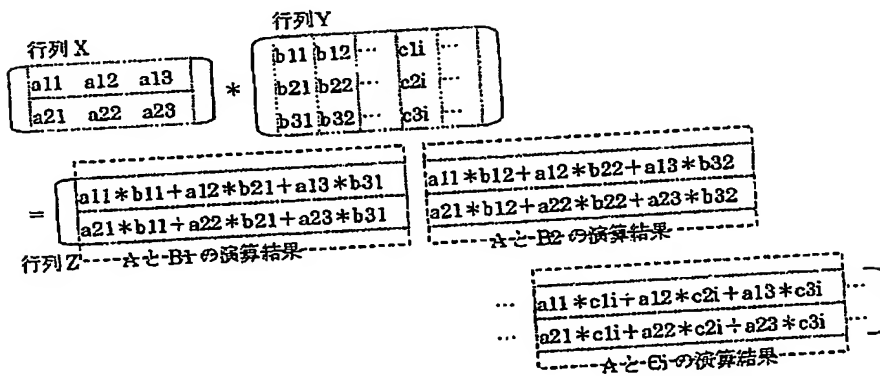
【図 6 2】



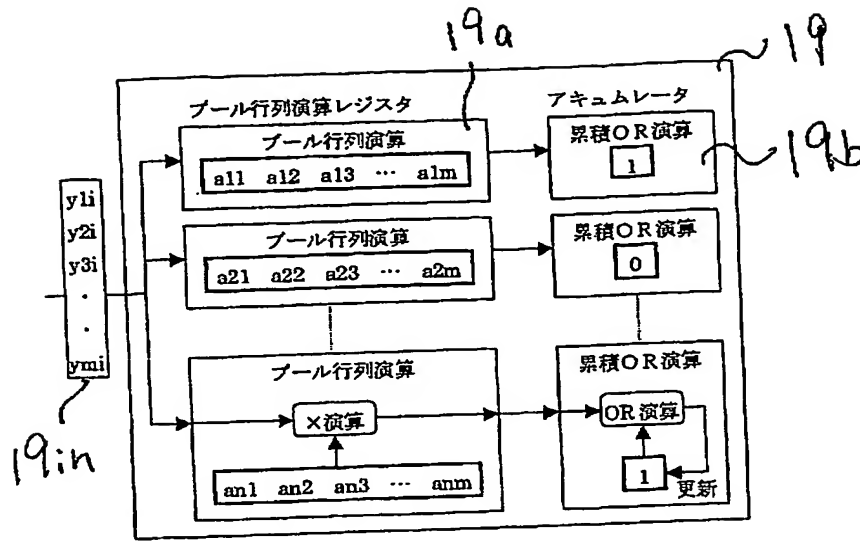
【図 6 3】



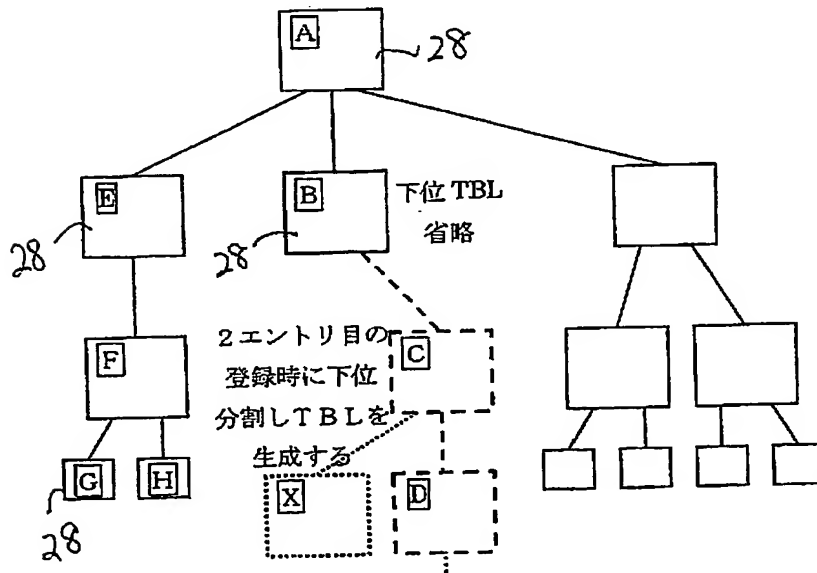
【図 6 4】



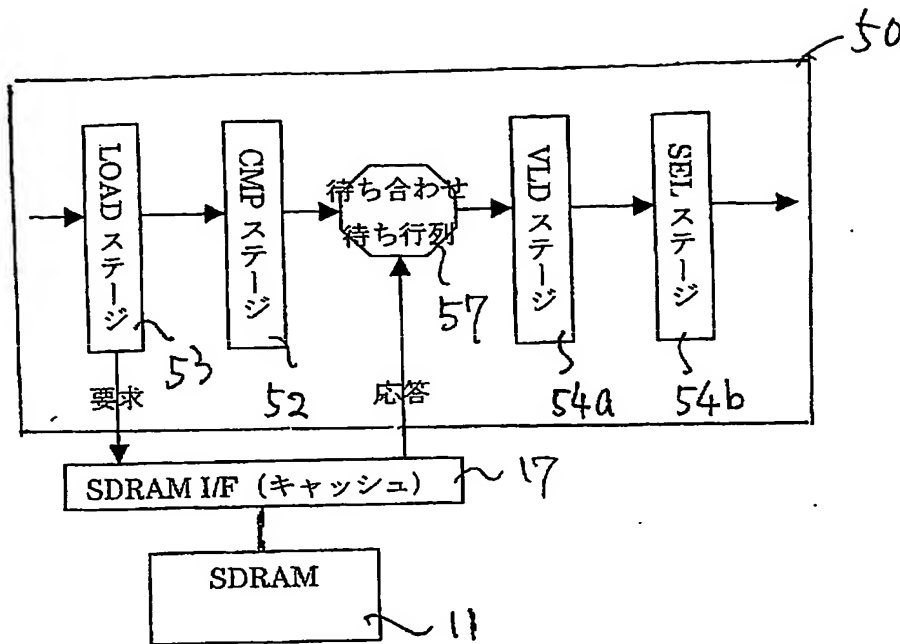
【図 6 5】



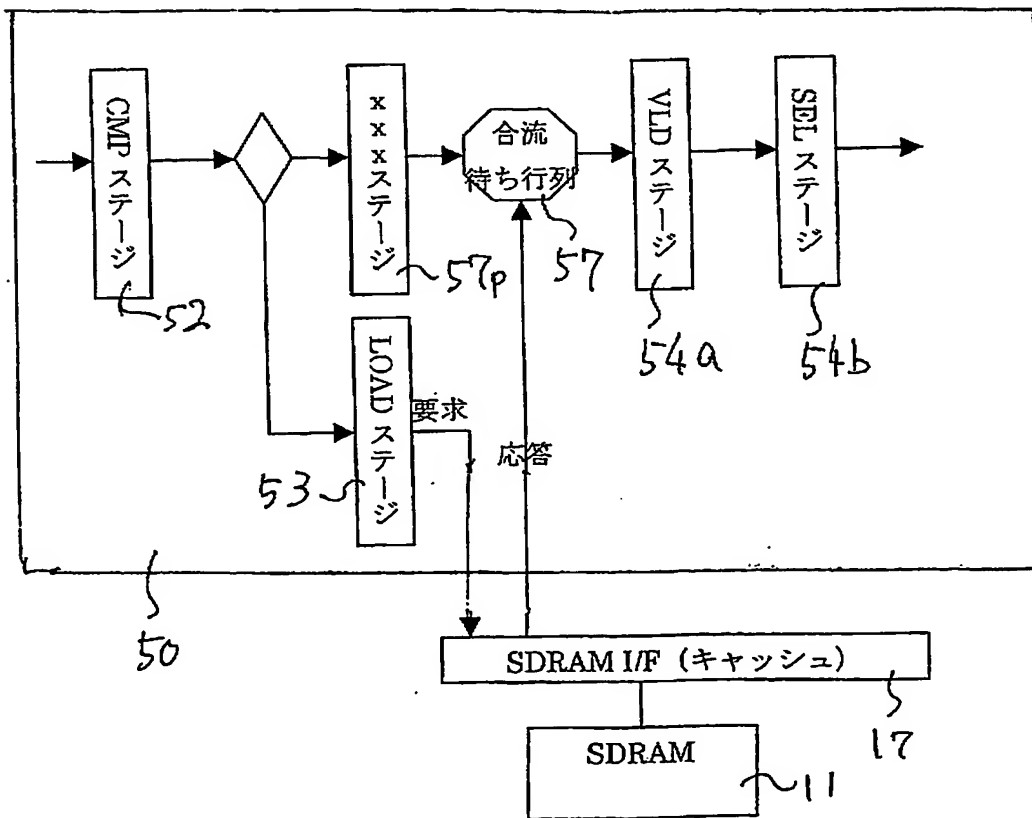
【図 6 6】



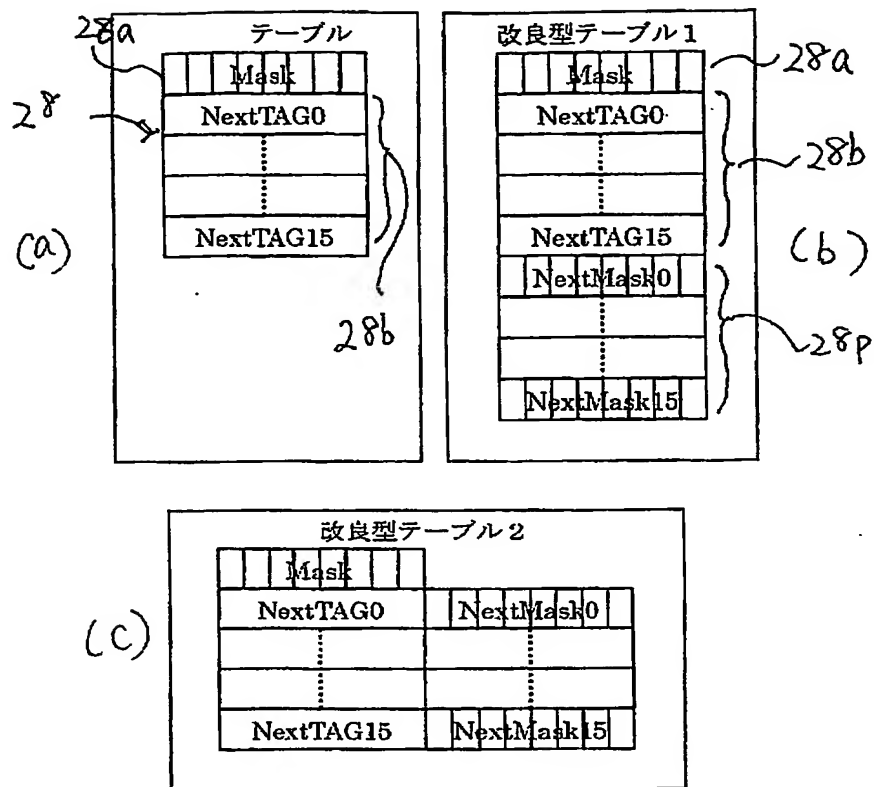
【図 6 7】



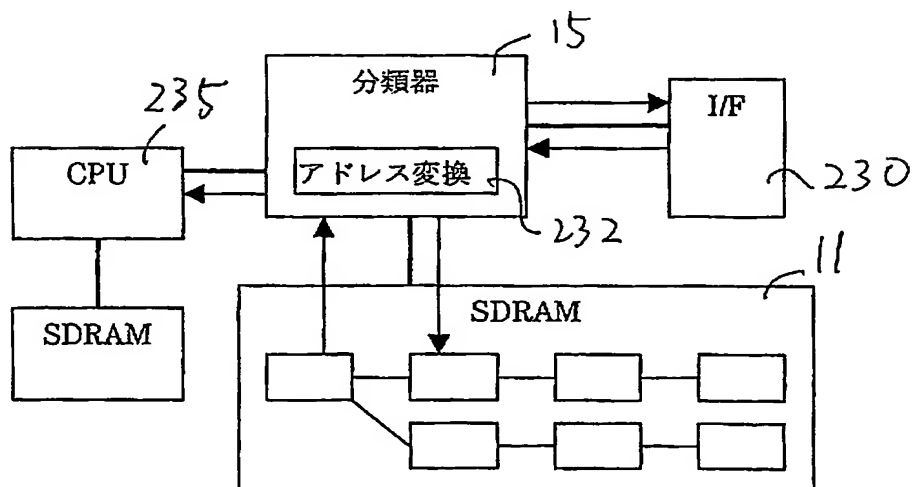
【図 6 8】



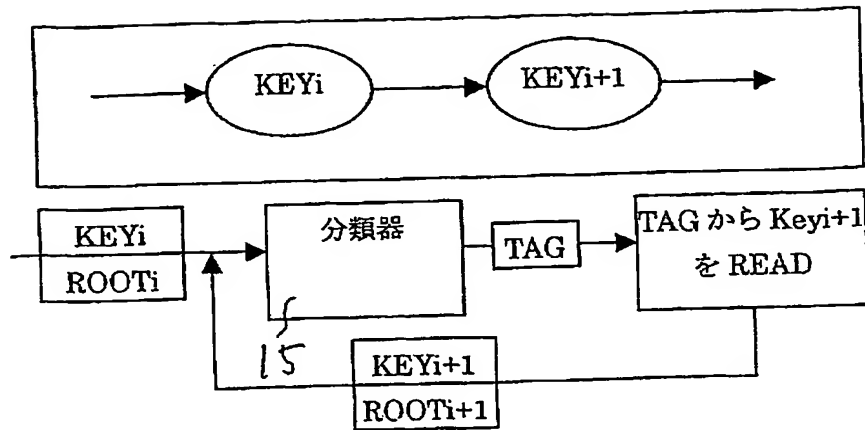
【図 69】



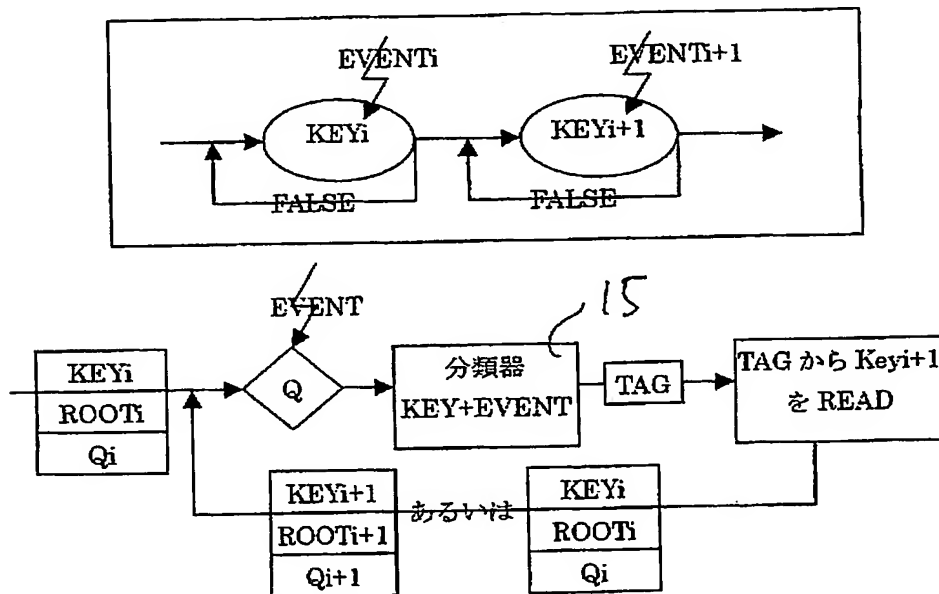
【図 70】



【図 7 1】



【図 7 2】



【書類名】 要約書

【要約】

【課題】 範囲一致を始め、完全一致および最長一致といった検索機能を提供可能であり、上位アプリケーションソフトウェアの負担が軽いビットストリングの照合装置を提供する。

【解決手段】 照合装置 50 においては、部分被検索キー 36 を比較ユニット 52 により取り得る全ての値と照合し、その照合段階のエントリを示すマスクデータ 28a を用いて判定ユニット 54 により部分被検索キー 36 の照合結果を得る。この照合装置 50 においては、照合テーブル 28 にエントリを追加削除する際にエントリ移動がなく、上位アプリケーションによるテーブル管理のオーバーヘッドを縮小または排除できる。

【選択図】 図 18

認定・付加情報

特許出願の番号	特願 2002-291708
受付番号	50201493492
書類名	特許願
担当官	第八担当上席 0097
作成日	平成14年10月 4日

<認定情報・付加情報>

【提出日】	平成14年10月 3日
-------	-------------

次頁無

【書類名】 出願人名義変更届 (一般承継)
【整理番号】 020320P531
【あて先】 特許庁長官殿
【事件の表示】
 【出願番号】 特願2002-291708
【承継人】
 【識別番号】 503253828
 【氏名又は名称】 株式会社インフォーエス
【承継人代理人】
 【識別番号】 100102934
 【弁理士】
 【氏名又は名称】 今井 彰
【提出物件の目録】
 【包括委任状番号】 0311781

認定・付加情報

特許出願の番号	特願 2002-291708
受付番号	50301629340
書類名	出願人名義変更届 (一般承継)
担当官	塩野 実 2151
作成日	平成15年11月18日

<認定情報・付加情報>

【提出日】 平成15年10月 1日

特願 2002-291708

出 願 人 履 歴 情 報

識別番号

[502254981]

1. 変更年月日

2002年 7月15日

[変更理由]

新規登録

住 所

東京都中央区日本橋浜町3-16-7

氏 名

有限会社アイシーエル

特願 2002-291708

出 願 人 履 歴 情 報

識別番号

[503253828]

1. 変更年月日

2003年 7月15日

[変更理由]

新規登録

住 所

東京都中央区日本橋浜町三丁目16番7号

氏 名

株式会社インフォーエス

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES.
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☒ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☒ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.